Петрів В.Ф., Ріпко Н.А.

Інформатика

АЛГО – основи програмування

8 клас

Навчальний посібник

Шепетівка
«ПП Шестопалов»
2008

УДК.004.451 (07)
ББК.32.973.26-018.2я7
П52

Рецензент: ***П. М. Зінько***, кандидат фізико-математичних наук,
доцент кафедри системного аналізу та теорії
прийняття рішень факультету кібернетики
КНУ ім. Тараса Шевченка.

Петрів В.Ф., Ріпко Н.А.

П52 Інформатика. АЛГО – основи програмування. 8 клас. / Навчальний посібник. – Шепетівка: «ПП Шестопалов», 2008. – 104 с.

ISBN 978-966-2017-22-9

Рекомендується для 8-х класів загальноосвітніх навчальних закладів різних профілів. Відповідає вимогам діючих програм з інформатики та орієнтований на використання 12-бальної шкали оцінювання знань учнів.

У посібнику описане середовище АЛГО, в якому реалізовані елементи мови Паскаль, необхідні початківцю для розуміння основ програмування. На простих прикладах розглядаються оператори розгалуження, варіанту, циклів, використання процедур та функцій. Матеріал посібника містить велику кількість задач та зразки завдань.

УДК.004.451 (07)
ББК.32.973.26-018.2я7

ISBN 978-966-2017-22-9

Передмова

Навчальний посібник рекомендується для 8-х класів загально-освітніх навчальних закладів різних профілів, розрахований на вивчення протягом 34 годин, орієнтований на практичне використання комп'ютерів на кожному уроці.

У посібнику описане середовище програмування АЛГО, в якому реалізована мова програмування Паскаль. Але вибрані тільки ті елементи мови, які необхідні початківцю для розуміння основ програмування.

Основні відмінності реалізованої мови від мови середовища Турбо Паскаль полягають у тому, що:

- не підтримуються вказівники, об'єкти, модулі;
- не реалізовані множинні, перелічувані та діапазонні типи;
- відсутній оператор with;
- підтримуються тільки текстові файли;
- дещо відрізняється система графічних процедур і функцій. Спосіб встановлення кольорів такий самий, як у всіх програмах, що працюють з використанням графічного інтерфейсу користувача.

Матеріал посібника розділений на параграфи, кожен з яких відповідає одному уроку і містить:

- викладення нового матеріалу, у якому теоретичний матеріал розглядається на прикладах розв'язування задач;
- питання для самоконтролю на закріплення теоретичного матеріалу (можуть бути використані для самостійної роботи);
- набір задач з даної теми, які можуть бути використані для роботи на уроці або як домашнє завдання. Деякі задачі позначені «*», що вказує на підвищений рівень складності.

В окремих параграфах подані зразки завдань для тематичного оцінювання. Посібник складений таким чином, що знання попередніх тем закріплюється та доповнюється наступним матеріалом.

Останній розділ «Приклади цікавих програм» є узагальненням матеріалу всього посібника. Кожен параграф цього розділу може використовуватись для проведення окремої комплексної практичної роботи (за часом відповідає 1-2 урокам), для виконання якої потрібні знання базових конструкцій мови програмування та вміння будувати графічні зображення.

Зміст

1. Базові конструкції мови програмування Паскаль

1.1. Середовище програмування АЛГО

Загальний вигляд та основні елементи оболонки

У середовищі АЛГО реалізована мова програмування Паскаль, яка була розроблена Н.Віртом у 1968–1970 роках спеціально для навчання програмуванню і одержала визнання завдяки наочності програм і легкості вивчення. У 1984 році на ринку програмних продуктів з'явилася система програмування Турбо Паскаль фірми Borland, що стало справжньою революцією у програмуванні. До цього під час вивчення програмування здебільшого використовували Бейсік – просту мову програмування для початківців, не пристосовану для розробки великих програм. Натомість Турбо Паскаль надавав зручне середовище, що забезпечувало роботу з текстом, відлагодження та запуск програм.

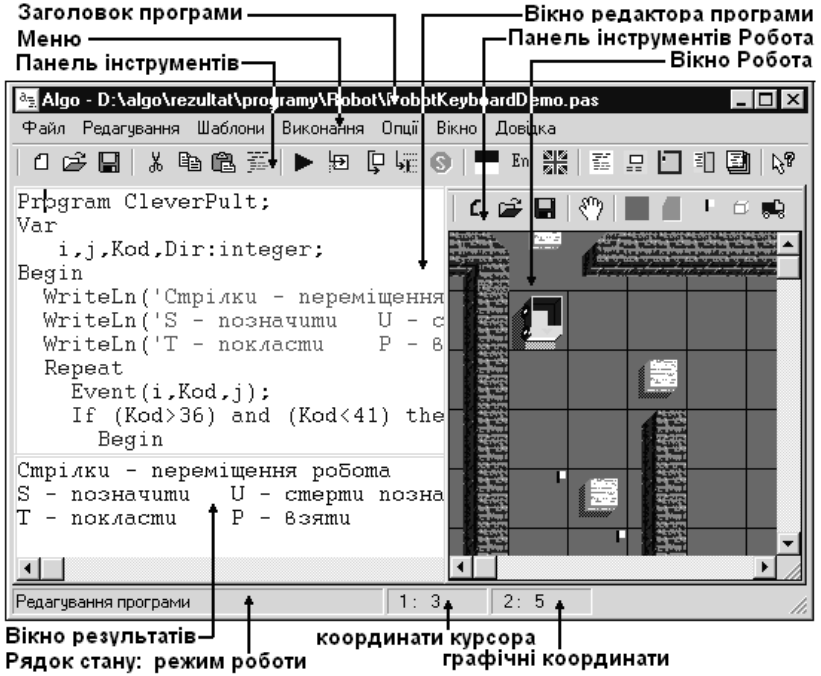
Для запуску розробленої програми необхідна спеціальна програма – компілятор, який перекладає цю програму мовою команд процесора, та текстовий файл, в якому записана сама програма. Щоб підготувати такий файл потрібно скористатись текстовим редактором. Крім того, потрібно мати можливість запустити програму на покрокове виконання, мати засоби для роботи з файлами, перегляду проміжних результатів та багато іншого. Всі ці засоби, зібрані разом, називають системою програмування, а елементи, які відображають на екрані виконувані дії (вікна, меню, кнопки) та забезпечують необхідні сервісні можливості для програміста, – середовищем програмування.

Загальний вигляд головного вікна системи програмування АЛГО наведений на малюнку (*див. наступну сторінку*).

АЛГО дозволяє відображати команди програми однією з двох мов – англійською або українською. Для переходу на англійську мову потрібно натиснути кнопку з англійським прапорцем, а на українську – з українським. Це саме можна зробити, натиснувши на клавіатурі **Ctrl+E** та **Ctrl+U** відповідно або командами меню (Опції / Програма / Англійською мовою чи Українською мовою).

На малюнку подане розгорнуте меню. Ліворуч від більшості команд показані відповідні кнопки панелі інструментів, а праворуч вказані клавіші для виклику команд з клавіатури («гарячі клавіші»). Для переходу до меню вибирають мишею відповідний його пункт

або натискають клавішу **Alt** і вибирають потрібний пункт клавішами керування курсором, після чого натискають **Enter**.



Робоче поле програми може містити три вікна. Їх можна по-різному розміщувати на екрані. Для цього використовують пункт меню **Вікно** або відповідні кнопки панелі інструментів. **Початківцям рекомендується використовувати режим поділу робочого поля**, щоб усі вікна було видно одночасно.

АЛГО надає можливість автоматичного впорядкування тексту програми (запису кожного оператора з нового рядка, відступів перед вкладеними операторами тощо). Для виконання цієї операції слід вибрати в меню команду **Редактор / Впорядкувати**.

Для більш швидкого набору програми забезпечена можливість вставляти в текст оператори, описи та інших конструкції, які можна вибирати з меню **Шаблони**.

Це ж меню можна активізувати клавішею **F10** або натисканням **правої** кнопки миші в режимі редагування програми. Вибраний текст вставляється в програму **на місце курсора**. Цим меню користуються також, щоб вибрати колір для команд.

Зчитування та запис програм

Для того, щоб прочитати (завантажити) програму з диска, потрібно вибрати команду меню **Файл/Прочитати**. При цьому буде відкрите стандартне вікно для вибору файлу, в якому слід вказати папку та шуканий файл.

Для того, щоб створити новий файл, потрібно вибрати пункт **Файл/Новий** в меню. Якщо поточна програма містить не збережені зміни, то буде виведене відповідне попередження.

Щоб зберегти файл слід вибрати команду меню **Файл / Записати**. Буде відкрите стандартне вікно системного діалогу для збереження файлу. Якщо файл вже був записаний і ви тільки вносили зміни, то система автоматично пропонує записати його з тим самим іменем, а зберігаючи новий файл, слід обов'язково ввести з клавіатури його ім'я у відповідному полі діалогу.

Якщо при цьому вказати розширення (.pas, .dat, .txt), то файл буде записаний з цим розширенням, інакше система автоматично запише файл з розширенням **.pas**.

Виконання та відлагодження програми

Щоб запустити програму на виконання потрібно вибрати команду меню **Команди / Виконати програму**. Розпочнеться компіляція, яка успішно завершиться лише тоді, коли в тексті програми немає синтаксичних помилок.

Виявивши помилку, компілятор припиняє роботу, рядок, в якому припинилася компіляція, виділяється червоним кольором і

курсор встановлюється на місці зупинки. Програміст може виправити помилку і знову запустити програму.

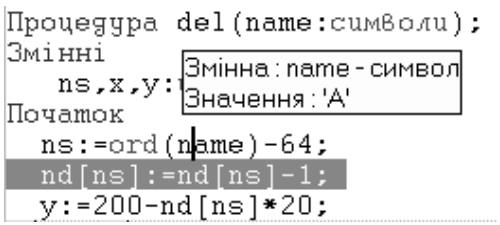
Якщо програма скомпілювалась, то розпочнеться її виконання. Але це не означає, що в ній зовсім немає помилок. Компілятор не може виявити логічних помилок (наприклад, якщо замість знака додавання написати знак множення). Кнопкою **Stop** або клавішею **F6** завжди можна припинити виконання програми.

Щоб відшукати логічні помилки у програмі, можна виконувати по одному оператору, для чого вибрати команду **Виконати наступний оператор** або **Трасувати програму (**клавіша **F7)**.

Щоб пропустити великий фрагмент програми або довгий цикл, потрібно встановити курсор на потрібному місці зупинки і скористатись командою меню **Виконання / Виконати до курсору**.

Ці ж команди можна подавати вказаними в меню функціональними клавішами або кнопками панелі інструментів.

Рядок програми, на якому припинилося виконання, буде позначений зеленим кольором, а курсор встановиться на початку оператора, який має виконуватися. Для перегляду значень змінних у цьому режимі достатньо навести вказівник миші на ім'я змінної і залишати його нерухомим одну секунду. Програма в маленькому жовтому віконечку виведе ім'я змінної, її тип та значення (*див. малюнок*).



Користування довідковою системою

Довідкова система середовища АЛГО побудована з використанням стандартних засобів операційної системи Windows.

При натисканні клавіші **F1** довідкова система активізується і демонструє розділ, в якому описане слово, на яке вказував курсор в момент натискання клавіші. Якщо такого розділу немає, то виводиться зміст довідкової системи. В цьому випадку потрібно відшукати у змісті потрібний розділ і вивести його, **двічі клацнувши мишею**.

Щоб запустити на виконання приклад програми, наведеної в довідковій системі, необхідно виконати такі дії:

- виділити текст програми (натиснувши ліву кнопку миші на початку програми, перемістити вказівник до кінця програми;

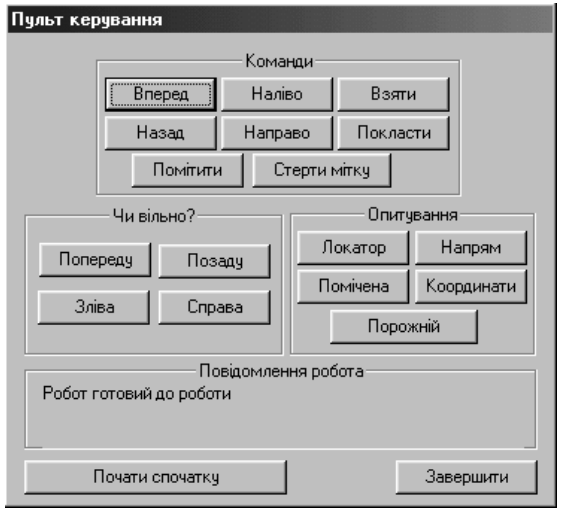
- помістити вказівник на виділений текст і натиснути праву кнопку миші та вибрати команду меню **Копіювати**;
- створити в АЛГО нову програму;
- вставити скопійований текст (команда меню **Редагування / Вставити**) та запустити програму.

Використання виконавця Робота

Щоб полегшити розуміння алгоритмічних конструкцій, в оболонку включений виконавець – робот-вантажник. Під час зчитування програми, що містить команди робота, відповідне вікно відкривається автоматично. В інших випадках це вікно потрібно відкривати командою меню **Опції / Робот / Використовується**. Операція **Опції / Робот / Не використовується** закриває вікно робота.

Робот працює в певній обстановці, схему якої можна редагувати, перейшовши до режиму редагування. Для цього користуються панеллю інструментів робота (*див. малюнок*). На активний режим роботи вказує змінена форма вказівника.

Підготовлену схему можна записати на диск для повторного використання. Рекомендується записувати схему з такою самою назвою файлу, як і програму, для якої вона підготовлена. В цьому випадку під час завантаження програми автоматично буде зчитана схема, підготовлена для цієї програми.



Щоб краще зрозуміти можливості Робота, можна скористатися режимом ручного керування.

Натискання кнопки з зображенням руки на панелі інструментів призводить до появи вікна діалогу ручного керування.

Клацаючи відповідні кнопки, Роботу посилають команди, які він виконує, або повідомляє про помилку.

Питання для самоконтролю:

1. *Опишіть загальний вигляд головного вікна програмного середовища АЛГО.*
2. *Прокоментуйте дію основних інструментів з панелі інструментів.*
3. *Як відкрити існуючу та зберегти створену програму?*
4. *За допомогою яких команд програма запускається на виконання? Які є режими виконання програми?*

Практичні вправи

1. Використовуючи меню **Шаблони**, наберіть текст програми:

```
Program NoName;
Var    i,j : integer;
Begin
  For i:=1 to 20 do
    Begin
      For j:=1 to 20 do
        Write( 8 );
      WriteLn;
    end
end.
```

– запустіть програму на виконання;

– перегляньте роботу програми в режимі покрокового виконання;

– змінюючи місце знаходження курсора та за допомогою клавіші **F1** дослідіть роботу контекстної довідки;

– замініть цифру 8 на іншу та спостерігайте, як зміниться результат роботи програми.

2. Відкрийте панель інструментів робота. Побудуйте лабіринт. За допомогою пульту керування виведіть робота з лабіринту.

1.2. Основні елементи мови програмування Паскаль

Алфавіт і словник мови програмування

При написанні програми використовують знаки, що утворюють алфавіт мови програмування:

- букви англійського алфавіту від **A** до **Z** і від **a** до **z**;
- букви українського алфавіту від **А** до **Я** і від **а** до **я** (тільки в АЛГО);

- арабські цифри від **0** до **9**;
- пропуск;
- спеціальні одиничні символи: **+ – * / = < > [ ] . , ' ( ) : ; { }**
- спеціальні пари символів: **<= >= (* *) <> ..**

Неподільні послідовності символів утворюють слова, що несуть у програмі певний зміст. Слова поділяються на зарезервовані та ідентифікатори об'єктів.

Зарезервовані (службові) слова є складовою частиною мови, мають фіксоване написання та раз і назавжди визначений зміст.

Наведемо таблицю зарезервованих слів мови Паскаль та їхніх перекладів, які використовуються під час роботи АЛГО українською мовою.

Зарезервовані слова мови Паскаль та їхні переклади

And	та	array	масив
Begin	Початок	case	Вибір
const	Стала	div	div
do	виконати	downto	назадДо
else	інакше	end	кінець
for	Для	function	функція
if	Якщо	label	Мітка
mod	mod	goto	ЙтиДо
or	або	not	не
program	Програма	of	із
repeat	Повторювати	procedure	Процедура
string	Рядок	record	Запис
then	то	type	Тип
until	докиНе	to	до
while	Поки	var	Змінна

Ідентифікатори (імена) використовують для позначення типів, констант, змінних, процедур і функцій, які вбудовані у систему програмування (стандартні) або визначені самим програмістом.

Ідентифікатор може мати довільну довжину, проте до уваги беруться лише перші 16 символів. Він не може починатися з цифри і не повинен містити пропусків. У ідентифікаторах допускаються букви, цифри і знак підкреслення «_».

Наведемо кілька прикладів ідентифікаторів:

G, alfa, test17, x2y, _h1,

Сума2Чисел, кінець_масиву.

Надалі слова «ім'я» та «ідентифікатор» вживатимемо як синоніми.

В АЛГО ідентифікатори стандартних типів, констант, функцій та процедур, виділяються в тексті програми червоним кольором.

При написанні зарезервованих слів та ідентифікаторів можна використовувати як великі, так і малі літери. Компілятор їх не розрізняє.

Правила оформлення програм

Програма починається із заголовка, що має такий вигляд:

```
Program <ім'я програми>;
```

Примітка. У цьому посібнику кутові дужки **<…>** означають: «тут слід написати…». В даному випадку замість **<ім'я програми>** у заголовку пишуть ідентифікатор програми *(див. приклади далі)*.

Після заголовка йдуть розділи описів, у яких повинні бути описані всі програмні об'єкти (константи, змінні, типи, процедури, функції, мітки), що будуть використані в програмі.

Після розділів описів йде розділ операторів, що починається зі службового слова **Begin** і закінчується службовим словом **End**, після якого ставиться крапка.

У цьому розділі задаються дії над об'єктами програми, оголошеними в розділі описів. Оператори в цьому розділі відокремлюються один від одного крапкою з комою. Так само відокремлюють один від одного розділи програми.

Приклад

```
Program Example;
Var a,b,Sum:integer;
Begin
      Writeln('a,b =');
      Readln (a,b);
      Sum:=a+b;
      Writeln('Sum =',sum);
End.
```

Ім'я цієї програми – **Example**. З розділів описів є лише один – розділ опису змінних. Він починається зі службового слова **Var**, після якого записують послідовність оголошень змінних, розділених крапкою з комою. У кожному оголошенні перераховуються через кому імена змінних одного типу, після чого ставиться двокрапка і вказується тип змінних. У даному прикладі описано три

змінні з ідентифікаторами **a**, **b** та **Sum**, всі вони мають тип integer, тобто значення змінних цього типу – цілі числа *(детальніше про типи даних буде далі)*.

Після розділу описів змінних іде розділ операторів. Він починається зі службового слова **Begin**, після якого йдуть оператори програми. Перший оператор – **Writeln ('a,b=')** – виклик стандартної процедури для виведення на екран тексту, що міститься між апострофами. Наступний оператор – **Readln(a,b)** – виклик стандартної процедури для читання даних з клавіатури. У даному випадку необхідно ввести два цілих числа через пропуск, тоді змінна **a** отримає значення, що дорівнює першому введеному числу, а змінна **b** – значення, що дорівнює другому введеному числу.

Наприклад, якщо ввести числа 10 і 20, то **a=10**, а **b=20**.

Після цих двох операторів стоїть оператор присвоєння: **Sum:=a+b** (:= – це знак оператора присвоєння).

Під час виконання цього оператора змінна **Sum** набуде значення, що дорівнює сумі чисел **a** і **b**. Оскільки в результаті додавання двох цілих чисел утворюється ціле число, то змінна Sum описана як ціла.

Наступний оператор – це знову оператор виведення **Writeln ('Sum=', Sum).** Він виведе на екран текст, розміщений між апострофами, а за ним – значення змінної **Sum**. В кінці розділу операторів стоїть службове слово **End**, після якого стоїть крапка.

Якщо в АЛГО перемкнути мову з англійської на українську, то ця сама програма матиме такий вигляд:

```
Програма Приклад;
Змінна  a,b,Sum:ціла;
Початок
  Вивести('a,b=');
  Ввести (a,b);
  Sum:=a+b;
  Вивести('Sum=',sum);
Кінець.
```

Надалі, в більшості прикладів ми будемо використовувати класичний запис програм (англійською мовою). При необхідності, для кращого розуміння тексту програм, можна користуватись автоматичним перекладом програми, змінюючи мову з англійської на українську кнопками панелі інструментів.

Питання для самоконтролю:

1. *З чого складається алфавіт і словник мови програмування?*
2. *Що таке зарезервовані слова?*
3. *Які правила запису ідентифікаторів?*
4. *З чого починається програма?*
5. *Як описуються змінні?*
6. *З чого починається розділ операторів?*
7. *Як записується оператор виведення?*
8. *Як записується оператор введення?*
9. *Як записується оператор присвоєння?*
10. *Чим закінчується програма?*
11. *Вкажіть правильні імена змінних:*

а) proba;	б) _fire_;	в) 5_step;
г) BoX;	д) fi_re;	е) step_5;
є) 5BoX;	ж) maMa;	з) _step_5.

12. *Які з наведених нижче послідовностей символів є операторами присвоєння:*

а) a:= b;	б) a = a+1;	в) a:b–sqr(2);
г) a * x + b := 0;	д) z:= 0;	е) z:= z+1;
є) z:= z + 1,2;	ж) y:= y;	з) –y:= y;

Задачі

1) *Змініть програму, наведену в параграфі, таким чином, щоб у ній обчислювався добуток двох чисел.*
2) *Скласти програму для обчислення периметру:*
   a). прямокутника, ширина й довжина якого вводяться з клавіатури;
   b). трикутника, довжини всіх сторін якого вводяться з клавіатури;
   c). довільного чотирикутника, довжини всіх сторін якого вводяться з клавіатури.
3) *Скласти програму для обчислення значення виразу:*
   $y=15x^2+8x-9$. Значення *x* ввести з клавіатури.

1.3. Складання найпростіших лінійних програм

Лінійними програмами називають такі програми, в яких команди виконуються послідовно одна за одною.

Правила запису математичних виразів

Майже в кожній програмі виконуватимуться обчислення, причому результати обчислень необхідно буде зберігати для

подальшого використання. Для цього існує оператор присвоєння, з яким ви ознайомились раніше. При його виконанні змінна, ім'я якої стоїть ліворуч від знаку ':=' отримує значення виразу, записаного праворуч. Яким би складним не був вираз, він має бути записаний в рядок. Слід дотримуватись правил запису арифметичних виразів:

- порядок виконання дій змінюють за допомогою круглих дужок. За відсутності дужок пріоритет математичних операцій звичайний: спочатку зліва направо виконуються множення і ділення, потім – додавання і віднімання;
- не можна опускати знак операції множення: 5ab ⇒ 5*a*b;
- звичайні дроби записуються в рядок (зверніть увагу на дужки!):

$$\frac{2a-5}{3+b} \Rightarrow (2*a-5)/(3+b);$$

- при необхідності у виразах використовуються стандартні функції або функції користувача *(див. далі)*;
- аргументи функцій записуються в круглих дужках.

Деякі стандартні математичні функції

Функція `sqr(x)` повертає квадрат значення аргументу, тобто sqr (x)=x^2=x*x.

Приклади

Математичний запис	Запис на Паскалі	Значення змінних	Результат
4^2	sqr(4)	–	16
x^2	sqr(x)	x=13	169
$(d+e)^2$	sqr(d+e)	d=2, e=5	49
$3^4=(3^2)^2$	sqr(sqr(x))	x=3	81

Функція `abs(x)` повертає абсолютну величину (модуль) значення аргументу.

Приклади

Математичний запис	Запис на Паскалі	Значення змінних	Результат
|12|	abs(12)	–	12
|–12|	abs(–12)	–	12
|x+y|	abs(x+y)	x=3, y=–5	2
|x|+|y|	abs(x)+abs(y)	x=3, y=–5	8

Стандартні процедури для введення та виведення даних

Більшість програм передбачають введення (наприклад, з клавіатури) користувачем певних даних та виведення результатів роботи (зокрема, на екран). Мова Паскаль надає всі необхідні засоби для реалізації введення та виведення у програмах. В попередньому параграфі для цього були використані процедури `Readln` та `Writeln.` Розглянемо детальніше роботу цих та інших процедур.

Введення даних забезпечується викликом процедур `Read` та `Readln`. Якщо передбачається введення декількох значень (наприклад, `Read(a,b,c)`), то їх можна ввести в одному рядку, відділяючи «пропуском», а в кінці натиснути **Enter**. Можна вводити кожне значення окремо, натискаючи щоразу **Enter**. Змінні одержують свої значення послідовно: спочатку `a`, потім `b`, і останньою – `c`. Введення даних з окремого рядка виконується за допомогою процедури `Readln.` Після зчитування останнього значення зі списку цієї процедури наступні дані будуть вводитись з початку нового рядка.

Для **виведення** повідомлень, значень змінних та виразів використовують процедури `Write` та `Writeln`. Дія процедури `Writeln` відрізняється тим, що після виведення курсор переводиться на новий рядок. Всі параметри процедури виведення розділяються комами.

Приклад при a=2, b=3

```
Writeln(a,b);              на екрані: 23
Writeln('a=',a);           на екрані: a=2
Writeln('a=',a,'b=',b);    на екрані: a=2b=3
Writeln('a+b=',a+b);       на екрані: a+b=5
```

Для одержання результатів у вигляді таблиць, колонок використовують форматоване виведення. При цьому після елемента списку виведення через двокрапку вказується кількість позицій на екрані для виведення її значення. Якщо позицій більше ніж потрібно, то вони заповнюються пропусками ліворуч від значення.

Приклад

```
Writeln(a:3,b:3);          на екрані: __2__3
Writeln('a+b=',a+b:4);     на екрані: a+b=___5
```

Процедура виведення `Writeln` без параметрів використовується для переведення курсора на новий рядок та виведення порожніх рядків.

Приклад

Скласти програму для обчислення значення виразу $y=|x^2-2|+3$. Значення змінної x ввести з клавіатури.

Розв'язування

```
Program Example;
Var x,y:integer;
Begin
  Write('x=');
  Readln (x);
  y:=abs(sqr(x)-2)+3;
  Writeln('y=',y);
End.
```

Питання для самоконтролю:

1. *Яких правил необхідно дотримуватись, записуючи арифметичні вирази мовою програмування?*
2. *Запишіть вираз $y=|x|$ мовою програмування Паскаль та знайдіть його значення при $x=-3$, $x=3$.*
3. *Запишіть вираз $z=|x-2|+3x^8$ мовою програмування Паскаль.*
4. *Запишіть вираз $a=6b^2+|b-3|^3-15$ мовою програмування Паскаль.*
5. *Запишіть вирази мовою програмування Паскаль:*

   а) $\frac{6x}{x-9}$; б) $\frac{x^2-2}{x+3}$; в) $\frac{3x}{4}+\frac{1}{x^2-1}$.

6. *Дано значення змінних x і y: x=14, y=3. Якими будуть значення цих змінних після виконання послідовності дій:*

   а) x:=y; y:=x;

   б) d:=x+1; x:=y; y:=d.

Записати у вигляді одного або декількох операторів присвоєння:

7. *Змінній x присвоїти значення, що дорівнює півсумі значень змінних x та y;*
8. *Подвоїти значення змінної a;*
9. *Значення змінної x збільшити на 0.1;*
10. *Змінити знак значення змінної t;*
11. *Поміняти місцями значення змінних x і y.*

Задачі

Написати програми для обчислення значення виразів:

1. $y=(3x^3+18x^2)*x+12x^2-5$;
2. $a=(d+c+b)*e-k-1$;
3. $d=3c^3+|c-4c+7|^3-5c$;

4. $c=|x+4|-|x-3x+6|$;
5. $y=5x^5-10x+2$;
6. $z=14x^4-5x^3+11x-17$.

1.4. Цілий і логічний типи даних. Умовний оператор

Прості типи даних

Описуючи змінну, необхідно зазначити її тип. Тип змінної визначає набір значень, яких вона може набувати, форму запису їх в пам'яті та операції, які можуть бути з нею виконані. Типи поділяються на прості та складні. Змінна простого типу завжди містить один елемент даних (число, літеру і т.п). Змінна складного типу являє собою таблицю значень одного типу (масив) або набір полів різних типів (запис). До простих типів в АЛГО належать:

- цілий тип **integer** – **ціла**;
- логічний тип **boolean** – **логічна**;
- символьний тип **char** – **літера**;
- дійсний тип **real** – **дійсна**.

Всі прості типи, крім дійсного, є **порядковими**. Всі можливі значення порядкового типу є впорядкованою множиною. Кожне можливе значення порядкового типу має номер, який є цілим числом. За винятком значень цілого типу, перше значення будь-якого порядкового типу має номер 0, наступне значення має порядковий номер 1 і так далі для кожного значення в цьому порядковому типі. Порядковим номером значення цілого типу є саме це значення. У будь-якому порядковому типі кожному значенню, крім першого, передує цілком визначене інше значення, і після кожного значення, крім останнього, слідує інше значення відповідно до упорядкованості типу.

Цілий тип даних

Змінна **цілого** типу в АЛГО може набувати значень з діапазону від –2147483648 до 2147483647 і займає в пам'яті 4 байти.

Приклад опису:

```
var a,k,D1,D2:integer;
```

До даних цілого типу можна застосовувати операції «+» – додавання, «–» – віднімання, «*» – множення, «/» – ділення і деякі інші.

Примітка. Оскільки при діленні одного цілого числа на інше не завжди виходить ціле число, то присвоювати результат операції ділення змінній цілого типу ***не можна***.

Натомість є дві операції, які застосовують тільки до даних цілого типу і отримують цілочисельний результат: **div** – ціла частина від ділення; **mod** – остача від ділення.

Приклад

19 div 4=4;	19 mod 4=3;
12 div 4=3;	12 mod 4=0;
–21 div 4= –5;	–21 mod 4= –1;
–7 div (–4)=1;	–7 mod (–4)= –3.

Логічний тип даних

При складанні програм, крім математичних можна обчислювати значення логічних виразів. Змінні **логічного** типу можуть набувати тільки одного з двох значень – False (*хибний* або *Ні*) і True (*істинний* або *Так*). Змінні логічного типу одержують значення в результаті виконання операцій порівняння (відношення): «<« (менше), «>« (більше), «<=« (менше або дорівнює) «>=« (більше або дорівнює), «<>« (не дорівнює), «=« (дорівнює). Результат операції відношення дорівнює True, якщо відношення задовольняється для значень операндів, що входять у нього, і False – у протилежному випадку.

Приклад опису змінних логічного типу:

```
var m1,m2,dd:boolean;
```

Логічний вираз може бути ***простим*** (наприклад, x>5) або ***складеним.*** Складені вирази утворюються з простих за допомогою логічних операцій ***and, or, not*** (і, або, не)**.**

Приклад
```
(x>=a) and (x<=b);
(x>a) or (x>b);
not (x>a);
```

Логічна операція **and** має істиний результат у тому випадку, коли обидва логічні вирази істині. Логічна операція **or** має істиний результат у тому випадку, коли істиний хоча б один логічний вираз. Логічна операція **not** завжди дає результат, протилежний значенню виразу.

Логічні операції, операції відношення й арифметичні операції часто зустрічаються в одному виразі. При цьому відношення, що стоять ліворуч і праворуч від знака логічної операції, потрібно взяти в дужки, оскільки логічні операції мають вищий пріоритет.

Арифметичні та логічні операції мають такий пріоритет:

not, – (унарний)
and, *, /, div, mod
or, xor, +, –
операції відношення.

Крім того, порядок виконання операцій регулюється дужками. У мові Паскаль не можна вводити логічні дані за допомогою оператора `Read`. Проте передбачено виведення значень змінних логічного типу за допомогою оператора `Write`.

Приклад

Обчислити значення виразу: `(a<b)and(b<c)and(a<c)`, при `a=1`, `b=2`, `c=3`. Значення виразу дорівнює `True`, тому що істиними є всі значення простих логічних виразів.

Умовний оператор

Якщо розв'язок задачі має кілька варіантів, що залежать від початкових умов, то при складанні програм використовуються умовні оператори. Вони забезпечують виконання чи не виконання оператора або блоку операторів залежно від заданих умов.

Умовний оператор має повну та скорочену форми.

Повний умовний оператор має вигляд:

```
If   <умова> Then <оператор 1>
             Else <оператор 2>;
```

або

```
Якщо <умова> To <оператор 1>
             Інакше <оператор 2>;
```

Виконання умовного оператора починається з обчислення значення логічного виразу, записаного в умові. Якщо умова істинна, то виконується <оператор1>, у протилежному випадку – <оператор 2>. Якщо на місці оператора треба записати серію операторів, то вони беруться в операторні дужки **Begin-End**. Декілька операторів, взяті в операторні дужки, називають ***складеним оператором***.

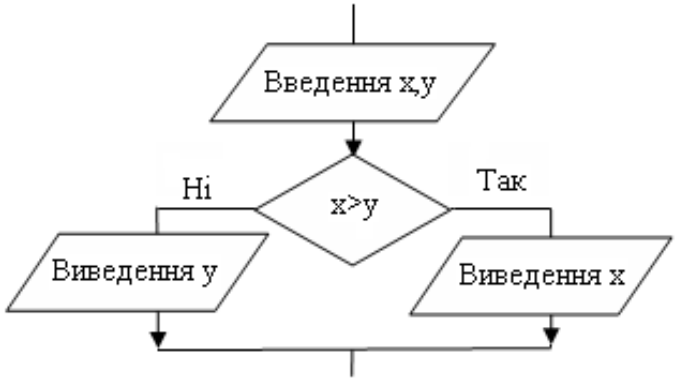
Приклад №1

Вивести на екран більше з двох даних чисел.

Розв'язування

```
Program Example;
Var x,y:integer;
```

```
Begin
  Write ('x,y=');
  Readln(x,y);
  If x>y Then Writeln (x)
         Else Writeln(y);
End.
```



Зверніть увагу на те, що перед службовим словом Else розділовий знак – крапка з комою – не ставиться.

Неповний умовний оператор має вигляд:

```
If  <умова>  Then <оператор>;
```

або

```
Якщо  <умова> To <оператор>;
```

Гілка Else може бути відсутньою, якщо у випадку невиконання умови нічого робити не потрібно. Наприклад, якщо значення змінної – x менше за 0, то потрібно замінити його на протилежне. Задача розв'язується за допомогою такого умовного оператора:

```
If  x<0  Then x:=-x;
```

Записуючи неповний умовний оператор слід бути уважним: якщо поставити після Then крапку з комою, програма скомпілюється, але працюватиме неправильно:

```
If  x<0  Then; x:=-x;
```

В такому випадку оператор x:=-x не є частиною умовного оператора, а буде виконаний обов'язково. Кажуть, що в гілці Then записаний ***порожній оператор***.

Приклад №2

Написати програму для перевірки, чи належить ціле число, введене з клавіатури, інтервалу [0,5].

Розв'язування

Позначимо через x число, яке вводиться з клавіатури користувачем. За умовою x – це змінна цілого типу. Число x належить заданому інтервалу [0, 5] лише в тому випадку, якщо одночасно виконуються дві умови: (x>=0) і (x<=5). Тому для утворення складної умови скористаємось логічною операцією **and**.

```
Program Example;
Var x:integer;
Begin
  Write ('x=');
  Readln(x);
  If(x>=0)and(x=<5) Then Writeln (x,' належить')
       Else Writeln(x,' не належить');
End.
```

Вкладені умовні оператори

Під час розв'язування задач часто розглядається не два, а більше варіантів. Це можна зробити, використовуючи послідовно кілька умовних операторів. У цьому випадку після службових слів **Then** і **Else** може записуватися новий умовний оператор.

Приклад №3

Дано цілі числа a,b,c. Якщо a<=b<=c, то всі числа замінити їх квадратами. Якщо a>b>c, то кожне число замінити найбільшим із них, у інших випадках – змінити знак кожного з чисел.

Розв'язування

Умову задачі перепишемо так:

a:=a^2, b:=b^2,c:=c^2, якщо a<=b<=c,

a:=c, b:=c, якщо a>b>c,

a:=–a, b:=–b, c:=–c – у решті випадків.

```
Program Example;
Var a,b,c:integer;
Begin
   Writeln('Введіть числа a,b,c');
   Readln(a,b,c);
   If(a<=b) and (b<=c)
```

```
      Then begin
             a:=SQR(a); b:=SQR(b); c:=SQR(c)
           end
      Else if (a>b) and (b>c) Then
           begin
             a:=c; b:=c
           end
      Else begin
             a:=–a; b:=–b; c:=–c
           end;
   Writeln(a:3,b:3,c:3)
End.
```

Примітка. Якщо вкладеними умовними операторами є неповні умовні оператори, то можуть виникати проблеми, пов'язані із встановленням меж умовних операторів. У таких випадках службове слово Else відноситься до найближчого If.

Завдання

У наведеній вище програмі змінимо умовний оператор таким чином:

```
If (a<=b) and (b<=c) then
   Begin
      a:=SQR(a);b:=SQR(b);c:=SQR(c);
      if (a>b) and (b>c) then
         begin a:=c;b:=c end
      Else begin a:=–a;b:=–b;c:=–c end;
   End;
```

Як зміниться хід виконання програми? Сформулюйте умову задачі, яку тепер розв'язує дана програма.

Питання для самоконтролю:

1. *Які типи належать до простих типів даних в АЛГО?*
2. *Яких значень можуть набувати змінні цілого типу та які операції з ними можна виконувати?*
3. *Визначити значення логічного виразу: (–3>=5) or not (7<9) and (0>3).*
4. *Обчислити значення виразів:*
   a). (a>5) and (b>5) and (a<20) and (b<30);
   b). not(a<15) or not (b<30);
   c). c or d and (b=20);

   при a=10, b=20, c=true, d=false.
5. *Записати послідовність операторів для знаходження неповної частки й остачі від ділення цілого числа a на ціле число b.*

6. *Якими будуть значення змінних j, k після виконання умовного оператора:*
   If j>k then j:=k–2 else k:=k–2;
   якщо початкові значення змінних дорівнюють:
   a). j=3, k=5;
   b). j=3, k=3;
   c). j=3, k=2.
7. *Є умовний оператор:*
   If d<>10 then writeln('ура!') else writeln('погано…!')
   Чи можна замінити його такими операторами, не порушивши логіки роботи програми:
   a). If d=10 then writeln ('ура!') else writeln('погано…!');
   b). If not (d=10) then writeln('ура!') else writeln('погано…!');
   c). If not (d=10) then writeln('погано!') else writeln('ура!');
   d). If not (d<>10) then writeln('погано!') else writeln('ура!');
8. *Після виконання операторів*

```
a:=0;
if a<>0 then; a:=2;
```

   значення змінної a дорівнює 2. Поясніть чому.
9. *Використовуючи складений оператор, спростіть такий фрагмент програми:*

```
If a>b then c:=1;
If a>b then d:=2;
If a<=b then c:=3;
If a<=b then d:=4;
```

10. *Яким буде значення змінної a після виконання операторів:*

```
a:=3;
if a<4 then Begin a:=a+2; a:=a+3 End.
```

11. *Запишіть умовний оператор, у якому значення змінної обчислюється за формулою a+b, якщо a – непарне, a*b, якщо a – парне.*
12. *Запишіть послідовність операторів для знаходження суми цифр заданого трицифрового числа.*

Задачі

1) *Вивести на екран номер координатної чверті, якій належить точка з координатами (x,y), за умови, що (x<>0) і (y<>0).*
2) *Написати фрагмент програми для підрахунку суми тільки додатних із трьох даних чисел.*
3) *Дано три числа. Написати програму для підрахунку кількості серед них чисел, що дорівнюють нулю.*

4) *Складіть програму, в якій із трьох введених із клавіатури чисел підносяться до квадрата додатні, а від'ємні залишаються без зміни.*
5) *Знайти найбільше з трьох даних чисел.*
6) *Дано три різних цілих числа, знайти середнє. Середнім назвемо число, більше від найменшого з даних чисел, але менше від найбільшого.*
7) *Якщо ціле число M ділиться без остачі на ціле число N, то вивести на екран частку від ділення, у протилежному випадку – вивести повідомлення, що M на N без остачі не ділиться.*
8) *Складіть програму для обчислення виразу:*
   a) max(x + y + z, x*z*y)+3;
   b) min (x^2+y^2, y^2+z^2)–4.

   Значення змінних x, y, z вводяться з клавіатури.
9) **Складіть програму для обчислення добутку двох більших із трьох введених із клавіатури чисел.*
10) **На числовій прямій проведено безліч відрізків однакової довжини L. Відстань між сусідніми відрізками H. Один з відрізків починається в точці з координатою X. Перевірити, чи належить введене число одному з відрізків.*

1.5. Оператор вибору

Оператор вибору (варіанту) можна розглядати, як узагальнення умовного оператора. Він дає змогу зробити вибір з декількох варіантів залежно від значення керуючої змінної. Формат запису оператора варіанту такий:

```
Case <порядкова змінна або вираз> of
   <константа 1>:<оператор 1>;
   <константа 2>:<оператор 2>;
   …
   <константа n>:<оператор n>;
[Else <оператор>; ]
End;
```

або

```
Вибір <порядкова змінна або вираз> із
   <константа 1>:<оператор 1>;
   <константа 2>:<оператор 2>;
   <константа n>:<оператор n>;
[інакше <оператор>; ]
Кінець;
```


Виконання оператора вибору починається з обчислення виразу, який повинен мати значення порядкового типу. У випадку, коли результат обчислення дорівнює одній з перелічених констант, виконується відповідний оператор. Потім керування передається за межі оператора вибору. Якщо значення виразу не збігається з жодною із констант, то виконується оператор, що стоїть після **Else**, якщо він є, або керування передається оператору, що слідує за **End**.

Примітка.

1. Тип кожної з констант повинен збігатися з типом виразу. Можна задавати не тільки одну константу, а й список констант *(див. приклад)*.
2. Гілка **Else** міститься у квадратних дужках. Це означає, що ця частина оператора вибору не обов'язкова.
3. У конструкції вибору (на відміну від умовного оператора) перед **Else** ставиться крапка з комою.
4. У якості операторів можуть використовуватися і складені оператори.

Приклад

Нехай при тестуванні учень отримав N балів з 20 можливих. Потрібно вивести суму балів з коротким коментарем.

Розв'язування

```
Program оцінка;
Var N :integer;
Begin
   Read(N);
   Case N of
     20:         WriteLn('Краще не буває!');
     19,18,17 :  WriteLn('Відмінно!');
     16,15,14,13:WriteLn('Добре.');
     12,11,10,9 :WriteLn('Задовільно.');
     8,7:WriteLn('Ще трохи, і все було б в порядку.');
   else
     WriteLn('Як нічого не знаєте, то хоча б щось вгадали!')
   end;
   WriteLn('сума балів – ',N:2,' з 20 можливих');
end.
```

Питання для самоконтролю:

1. *В яких випадках використовується оператор вибору?*
2. *Який загальний формат запису оператора вибору?*
3. *Яких правил потрібно дотримуватись, використовуючи оператор вибору при розв'язуванні задач?*

Задачі

1) *Скласти програму, для визначення пори року за номером місяця. Перевірити коректність введених даних (якщо номер місяця не належить проміжку від 1 до 12 – вивести повідомлення про помилку).*
2) *Скласти програму, при виконанні якої, за номером дня тижня виводиться повідомлення про те, який цей день: робочий чи вихідний.*
3) *Скласти програму, при виконанні якої за номером дня тижня виводиться його назва.*

1.6. Підготовка до оцінювання з тем «Створення лінійних програм» та «Організація розгалужень»

1. Вкажіть правильно записані ідентифікатори:
   а) 1xy; б) digit1; в) mas2;
   г) a$; д) begin; е) –AB.

2. Вкажіть правильно записані оператори введення значень змінних:
   а) read(x, y, z); б) read x, y, z; в) read(x);
   г) x := read(x); д) read(a; b); е) read(a, b+c);

3. Вкажіть правильно записані оператори виведення значень змінних:
   а) write(x, y); б) write x, y, z; в) print x;
   г) write(100); д) read(a; b); е) write(x, x+10,2);

4. Для виразів з лівого стовпчика виберіть правильну відповідь у правому стовпчику:

1) 8 div 3	а) 0
2) –13 div 2	б) –2
3) 2 div 5	в) 5
4) –13 mod 5	г) 2
5) –11 mod 3	д) 4
6) 5 mod 6	е) –3
	ж) –6

5. Записати мовою Паскаль вираз: y=|x–4|+12.
6. Які з операторів записані правильно:
   a). if a:=0 then a>0;
   b). if a>0 then a:=0;
   c). if 2*2=5 then WriteLn(2*2).
7. Які з операторів записані правильно:
   a). if x><0 then y:=x–2 else y:=x+2;
   b). if x=0 then y:=1024 else y:=x–1024;
   c). if (x=0) or (y<>0) then y:=x;
   d). if x=0 and a>0 then b:=a else b:=x;
8. Записати мовою Паскаль: якщо число є парним, то вивести «Так», інакше – вивести «Ні».
9. Записати мовою Паскаль формулу:

$$\begin{cases} -5, \textit{ якщо } x < 5; \\ x, \textit{ якщо } -5 \le x \le 0; \\ 2x, \textit{ якщо } 0 < x \le 3; \\ 6, \textit{ якщо } x > 3. \end{cases}$$

Практичні завдання

10) Скласти програму, при виконанні якої, за номером дня тижня виводиться повідомлення про кількість уроків у цей день.
11) Дано значення змінних a, b, c, які є довжинами трьох відрізків. Написати програму, що визначає, чи можна побудувати трикутник з відрізків заданої довжини.
12) Дано натуральне число N (N<=100), яке позначає вік людини. Додати до цього числа одне зі слів: «рік», «роки», «років», відповідно до норм української мови. Наприклад: 1 рік, 12 років, 52 роки.

1.7. Цикл із параметром

При складанні програм часто виникає необхідність багато разів повторити однин і той самий набір команд. У таких випадках застосовуються ***циклічні оператори***, а команди, що повторюються, називають ***тілом циклу***. Залежно від того, чи відома заздалегідь кількість повторень, розрізняють ***цикл з параметром*** та ***цикли з умовою***.

Оператор циклу з параметром застосовують тоді, коли заздалегідь відоме число повторень певної послідовності операторів. Для підрахунку кількості повторень вводиться змінна-параметр одного з порядкових типів (integer, boolean, char тощо). Є дві форми запису циклу з параметром:

1. **For** <Параметр>:=A **to** B **do** <Тіло циклу>;
   Для <Параметр>:=A **до** B **виконати** <Тіло циклу>;

2. **For** <Параметр>:=A **downto** B **do** <Тіло циклу>;
   Для <Параметр>:=A **назадДо** B **виконати** <Тіло циклу>;

Де A – початкове значення параметра, B – кінцеве значення параметра, тіло циклу – оператор (простий або складений). Початкове й кінцеве значення параметра циклу можуть бути подані константами, змінними або виразами відповідного типу.

Розглянемо, як виконується оператор циклу з параметром виду

For <Параметр>:=A **to** B **do** <тіло циклу>;

Спочатку обчислюються значення виразів A і B. Якщо A≤B, то змінна-параметр послідовно набуває значень рівних A, A+1,…,B–1, B (тобто з кроком 1) і для кожного з цих значень виконується тіло циклу. Якщо на початку A>B, то тіло циклу не буде виконане жодного разу.

У випадку, коли параметр циклу потрібно зменшувати, використовується друга форма оператора із службовим словом **downto.** Цикл виконується так само, але значення параметра змінюється з кроком, що дорівнює –1.

Якщо потрібно повторити кілька операторів, то вони беруться в операторні дужки **Begin–End** (складений оператор).

По завершенні виконання оператора циклу з параметром значення змінної-параметра вважається невизначеним.

Приклад №1

З чисел від 10 до 99 вивести ті, сума цифр яких дорівнює N (0<N≤18).

Розв'язування

Позначимо через ***k*** чергове число, ***p1*** – старшу цифру числа ***k***, ***p2*** – меншу цифру числа ***k***, ***S*** – суму. Число ***k*** будемо друкувати лише в тому випадку, коли сума ***p1*** і ***p2*** дорівнюватиме ***S***.

```
Program Example;
Var k,N,p1,p2,S:integer;
Begin
   Write('N=');
   Readln (N);
    For k:=10 to 99 do
```



```
      Begin
       p1:=k div 10;{виділяємо старшу цифру}
       p2:=k mod 10;{виділяємо молодшу цифру}
       S:=p1+p2;    {знаходимо суму цифр}
       If S=N then writeln(k)
      End
End.
```

У цій програмі цикл можна було записати коротше:

```
For k:=10 to 99 do
   If k div 10+k mod 10=N then writeln(k);
```

Проаналізуйте його роботу самостійно.

Приклад №2

Знайти всі двоцифрові числа, що діляться на N або містять цифру N.

Розв'язування

Якщо двоцифрове число задовольняє умову задачі, то для нього виконується хоча б одна з трьох умов: перша цифра дорівнює N (p1=n) або друга цифра дорівнює N (p2=n), або саме число ділиться на N (k mod n = 0).

Яку логічну операцію необхідно використати для об'єднання цих простих умов у складену?

Питання для самоконтролю:

1. *В яких випадках використовуються циклічні оператори?*
2. *Які особливості запису циклу з параметром?*
3. *До якого типу даних належить змінна параметру циклу?*
4. *В яких випадках при складанні циклу використовуються операторні дужки?*

Скільки разів буде виконано тіло циклу в наступних фрагментах програм (по можливост, перевірте на комп'ютері):

5. *For k:=–1 to 1 do …*
6. *For k:=10 to 20 do…*
7. *For k:=20 to 10 do…*
8. *k:=5; r:=15;*
   For i:=k+1 to r–1 do…
9. *k:=5; r:=15;*
   *For i:=0 to k*r do…*
10. *k:=r;*
    For i:=k to r do…
11. *Визначити значення змінної S після виконання таких операторів:*
    S:=0; N:=10;
    For i:=2 to N do S:=S+100 div i;

12. Перевірте роботу даної програми на комп'ютері. Проаналізуйте використання циклу For:

```
Program Demo;
Var c:boolean;
Begin
  For c:= false to true do
   writeln(c);
End.
```

Задачі

1) *Визначити кількість трицифрових натуральних чисел, сума цифр яких дорівнює заданому числу N.*
2) *Знайти суму натуральних непарних чисел, що менші за 100.*
3) *Знайти суму цілих додатних чисел із проміжку від A до B, що кратні 4 (значення змінних A і B вводяться з клавіатури).*
4) *Знайти суму цілих додатних чисел, що більші за 20, менші за 100, кратні 3 і закінчуються на 2, 4 або 8.*
5) **Скласти програму для піднесення до квадрату натурального числа, використовуючи таку закономірність:*

$1^2=1$
$2^2=1+3$
$3^2=1+3+5$
$4^2=1+3+5+7$
……
$N^2=1+3+5+7+9+\ldots+2(N-1)$

1.8. Задачі з використанням циклу з параметром

Дільники числа

Поняття подільності числа A на деяке натуральне число B уперше зустрічається в «Арифметиці...» Л.Ф.Магницького, написаній в 1703 році. Найпростіший спосіб знайти всі дільники числа A – перевірити по черзі подільність його на кожне з чисел 1,2,...,A. Для цього можна скористатися умовою:

```
A mod B = 0.
```

Легко перевірити, що в інтервалі від (A div 2) + 1 до A-1 дільників числа A немає. Тому дільники натурального числа

A лежать в межах від 1 до A div 2. Отже, фрагмент програми для **знаходження дільників числа A** може мати вигляд:

```
Write ('Задайте ціле число:');
Readln (A);
For B:=1  to  A div 2   do
 If A mod B=0 Then Writeln (B,'-дільник числа ',A);
```

Для **підрахунку кількості дільників** використаємо лічильник K, який буде збільшувати своє значення на 1 при виконанні умови подільності (K:=K+1).

Також не складно переробити програму для **знаходження суми дільників** числа A. При виконанні умови подільності знайдений дільник B додається до змінної S, яка зберігає поточне значення суми (S:=S+B). Перед початком циклу змінну S потрібно зробити рівною 0.

Прості, досконалі та дружні числа

Натуральне число A, більше від одиниці, називається ***простим,*** якщо воно ділиться лише на одиницю і саме на себе.

Досить підрахувати кількість дільників заданого числа та проаналізувати результат: якщо K=2, то число просте. Ще простіше визначити, чи виконувалась хоч раз умова подільності для «претендентів на звання дільника» – чисел від 2 до A div 2:

```
Write ('Задайте ціле число:');
Readln (A);
K:=0;
For B:=2 to A div 2 do
   If A mod B=0  Then K:=K+1;
If K=0 Then Writeln (A,' - просте число.')
      Else Writeln (A,' - складене число.');
```

Деякі властивості простих чисел ще не достатньо вивчені. Це спонукало Г.Вейля до такого визначення:

«Прості числа – це такі істоти, які завжди схильні ховатися від дослідника».

Необхідно зазначити, що всі прості числа, крім 2, непарні. З цією категорією чисел пов'язане таке поняття, як **числа-близнята,** так називають два непарних простих числа, які відрізняються на 2.

Досконалими числами називають числа, рівні сумі своїх правильних (тобто менших від цього числа) дільників.

Стародавнім грекам були відомі лише чотири досконалих числа: 6, 28, 496, 8128. Видатний грецький філософ і математик

Нікомах із Гераси (I ст. н.е.) писав: **«Досконалі числа прекрасні. Проте відомо, що прекрасні речі зустрічаються рідко, а поганих усюди достатньо».**

В XII ст. церква запевняла, що для "спасіння душі" достатньо знайти п'яте досконале число. Воно було знайдене лише в XV столітті. До цього часу не знайдено жодного непарного досконалого числа, але й не доведено, що його не існує.

Скориставшись алгоритмом обчислення суми дільників, можна визначити, чи є число досконалим (наприклад: 6=1+2+3; 28=1+2+4+7+14) :

```
Write ('Задайте ціле число:');
Readln (A);
S:=1;
For B:=2  to  A div 2  do
   If A mod B=0  Then S:=S+B;
If S=A Then Writeln (A,'- досконале число.')
      Else Writeln (A,'- не досконале число.');
```

Проте, для «спасіння душі», необхідно скористатися додатковим циклом, тобто вкласти вищевказаний алгоритм у цикл, параметр якого рахує кількість знайдених досконалих чисел.

Дружніми називаються два натуральних числа, кожне з яких дорівнює сумі правильних дільників іншого. Ця назва походить з легенди, в якій розповідається, що коли Піфагора запитали, що таке дружба, він відповів: ***«220 та 284».*** Можна дати й інше визначення дружніх чисел: сума всіх дільників одного і другого такого числа рівна дорівнює обох чисел.

Питання для самоконтролю:

1. *Як знайти дільники заданого натурального числа A?*
2. *Яку функцію використовують, щоб визначити, чи ділиться число A на число B без остачі?*
3. *Які числа називають простими. Наведіть приклади?*
4. *Які числа називають досконалими. Наведіть приклади?*
5. *Які числа називають дружніми. Наведіть приклади?*

Задачі

1) *Написати програму для знаходження всіх дільників заданого натурального числа A.*
2) *Написати програму для знаходження кількості дільників заданого натурального числа A.*

3) *Написати програму для знаходження суми дільників заданого натурального числа A.*
4) *Написати програму для перевірки, чи є задане натуральне число A простим.*
5) *Написати програму для перевірки, чи є задане натуральне число A досконалим.*
6) *Натуральне число з n цифр називається **числом Армстронга**, якщо сума його цифр, піднесених до n-го степеня, дорівнює самому числу (наприклад, $153=1^3+5^3+3^3$). Відшукати всі числа Армстронга, що складаються з 3-х цифр.*

1.9. Цикл з передумовою

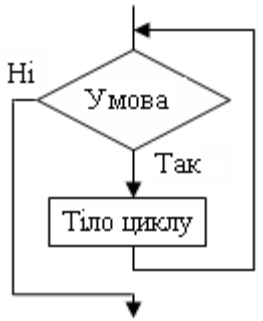
Тоді, коли число повторень тіла циклу заздалегідь невідоме, а задається лише деяка умова виконання циклу, використовуються ***цикли з умовою***, а саме цикл з передумовою та цикл з післяумовою. У цьому параграфі розглянемо перший з них.

Оператор циклу з передумовою має вигляд:

While <умова> **Do** <тіло циклу>

або

Поки <умова> **виконати** <тіло циклу>



Виконання оператора циклу з передумовою починається з перевірки умови, записаної після слова While. Якщо вона виконується, то виконується тіло циклу, потім знову перевіряється умова і т.д. Якщо під час чергової перевірки з'ясується, що умова не виконується, то тіло циклу виконуватися не буде. Керування перейде до оператора, записаного після циклу.

Примітка

1. Якщо тіло циклу складається з кількох операторів, то вони об'єднуються операторними дужками **Begin–End**.
2. У тілі циклу обов'язково має бути оператор, що впливає на істинність умови, інакше станеться зациклювання: оператори тіла циклу будуть повторюватися «вічно».

Приклад №1

Підрахувати кількість цифр заданого натурального числа n.

Розв'язування

Раніше ми виділяли цифри двоцифрових та трицифрових чисел. В цьому ж випадку ми не знаємо скільки цифр має число. Тому, поки число не стане рівне 0, будемо виконувати таку послідовність команд: збільшувати лічильник кількості цифр числа на одиницю, а число зменшувати в 10 разів (за допомогою цілочисельного ділення позбавлятимемось останньої цифри числа).

```
Program Example;
Var m, n, k: integer;
Begin
   Write('Введіть натуральне число:');
   Readln(n);
   m:=n; {копіюємо введене число}
   k:=0; {змінна-лічильник кількості цифр}
   While m<>0 Do
   Begin
      k:=k+1; {збільшуємо лічильник цифр}
      m:=m div 10 {відкидаємо останню цифру}
   End;
   Writeln('У числі ', n,'-', k, ' цифр')
End.
```

Роботу цієї програми цікаво спостерігати в покроковому режимі виконання (трасуванні), спостерігаючи зміну значень m і k.

Приклад №2

Підрахувати суму цифр заданого натурального числа n.

Розв'язування

Щоб розв'язати цю задачу, досить зробити незначні зміни в попередній програмі. Потрібно, як і раніше, відділяти останню цифру числа, але перед цим її потрібно запам'ятовувати в додаткову змінну (наприклад, a) і додавати до суми S.

Фрагмент програми:

```
While m<>0 Do
Begin
     a:=m mod 10;
     s:=s+a;
     m:=m div 10
End;
```


Питання для самоконтролю:

1. *В яких випадках використовується цикл з передумовою та які особливості його запису?*
2. *У даному фрагменті програми обчислення кількості цифр числа a знайдіть помилку та виправте її.*

```
ck:=0;
While a>=0 Do
Begin
   ck:=ck+1;
   a:=a div 10
End;
```

3. *Дано послідовність операторів:*

```
a:=1; b:=1;
While a+b<8 Do
  Begin a:=a+1; b:=b+2 End;
s:=a+b;
```

Скільки разів буде повторене тіло циклу? Якими будуть значення змінних a, b і s після виконання цієї послідовності операторів?

4. *Якими будуть значення змінних a і b після виконання послідовності операторів:*

```
a:=1; b:=1;
while a<=3 Do a:=a+1; b:=b+1;
```

5. *Знайдіть значення змінної s після виконання таких операторів:*

```
а)s:=0; i:=0;
   While i<5 Do i:=i+1; s:=s+100 div i;
б)s:=0; i:=0;
   While i>1 Do
     Begin s:=s+100 div i; i:=i-1 End;
```

6. *Яким умовам повинно задовольняти значення змінної k, щоб такі цикли були нескінченними:*

```
а)While c<0 Do c:=c+k;
б)While k<>0 Do k:=k+1;
в)While k<>0 Do k:=k+2;
```

Задачі

1) *Знайти кількість парних цифр натурального числа.*
2) *Скільки разів дана цифра зустрічається у цілому числі?*
3) *Знайдіть найбільшу цифру цілого числа.*
4) *Знайдіть старшу цифру числа.*

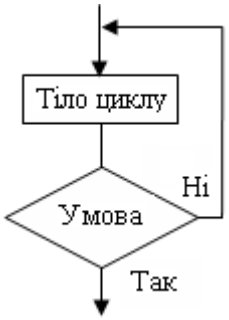
5) *Поміняти першу і останню цифри числа місцями.*
6) *Скласти програму для перевірки, чи є задане натуральне число паліндромом.*

1.10. Цикл з післяумовою

Для програмної реалізації циклічних алгоритмів із невідомим числом повторень є ще один оператор – оператор циклу з післяумовою, що має такий вигляд:

Repeat < оператори >; **Until**<умова зупинки циклу>;	**Повторювати** < оператори >; **ДокиНе**<умова зупинки циклу>;

Цей оператор відрізняється від циклу з передумовою тим, що перевірка умови проводиться після чергового виконання тіла циклу. **Це забезпечує виконання тіла циклу хоча б один раз.**



Зверніть увагу на те, що даний оператор циклу допускає наявність кількох операторів у тілі циклу, тому службові слова **Begin** і **End** не потрібні.

Порядок виконання циклу з післяумовою такий: виконується послідовність операторів, що складають тіло циклу, після чого перевіряється умова, записана після службового слова `Until`. Якщо умова виконується, то цикл завершується. У протилежному випадку оператори тіла циклу виконуються ще раз, після чого знову перевіряється виконання умови.

Приклад

Скласти програму для планування купівлі в магазині товарів на суму, що не перевищує задану величину.

Розв'язування

Позначимо через `x` та `k` ціну та кількість товару, через `p` – задану граничну суму, через `s` – вартість покупки. Початкове значення загальної вартості покупки `s` дорівнює нулю. Значення граничної суми вводиться з клавіатури. Необхідно повторювати запит ціни й кількості вибраного товару, визначати його вартість,

додавати її до загальної вартості та виводити результат на екран доти, поки вартість не перевищить граничну суму `p`.

```
Program  Exemple;
Var    x,k,p,s:integer;
Begin
 Write('Гранична сума--');
 Readln(P);
 S:=0;
 Repeat
   Write('Введіть ціну товару та його кількість:');
   Readln(x,k);
   S:=s+k*x;
   Writeln(' вартість покупки дорівнює ',s)
 Until s>p;
Writeln('вартість покупки перевищила граничну суму!');
End.
```

Питання для самоконтролю:

1. *В яких випадках використовується цикл з післяумовою та які особливості його запису?*
2. *У чому подібність і відмінність циклів з умовами?*
3. *Виберіть правильну відповідь.*

При виконанні оператора повторення *Repeat*

а) спочатку обчислюється логічний вираз, і в залежності від результату виконуються або не виконуються оператори тіла циклу;

б) спочатку виконуються оператори тіла циклу, потім обчислюється логічний вираз, результат якого впливає на повторне виконання операторів.

4. *Визначити значення змінної s після виконання таких операторів:*

```
s:=0; i:=1;
Repeat
  s:=s+5 div i;
  i:=i-1
Until i<=1;
```

5. *Що буде надруковано в результаті виконання такої послідовності операторів:*

```
i:=1;
repeat
     write(i,' ');
     i:=i+2
until i>19;
```

6. *Визначити значення змінних s та i після виконання таких операторів:*

```
S:=0; i:=1;
repeat
     S:=S+i;
     i:=i+1
until i>10;
```

1.11. Підготовка до оцінювання теми «Циклічні конструкції»

1) *У заданих фрагментах програм обчислюється сума натуральних чисел з інтервалу від 1 до N. Прокоментуйте особливості використання кожного циклу.*

```
S:=0;
for i:=1 to N do
    S:=S+i;
writeln('S=',S);
```

```
i:=1;
S:=0;
while i<=N do
 begin
 S:=S+i;
 i:=i+1
 end;
writeln('S=',S);
```

```
i:=1;
S:=0;
repeat
 S:=S+i;
 i:=i+1
until
i>N;
```

2) *Проаналізуйте роботу даних фрагментів програм. Яке значення матиме змінна a в кожному випадку при p=12345. Сформулюйте умову задачі до кожного фрагменту.*

```
While p<>0 Do
Begin
    a:=a+p mod 10;
    p:=p div 10
End;
```

```
While p<>0 Do
Begin
    a:=a*10+p mod 10;
    p:=p div 10
End;
```

3) *Вказати значення змінної S, якого вона набуде після виконання таких операторів:*

1) s:=2; i:=0;
while i<5 do i:=i+1; s:=s+1/i; _____
2) s:=5; i:=1;
while i>1 do begin s:=s+1/i; i:=i–1 end; _____
3) s:=1; i:=1;
while i<4 do begin i:=i+1; s:=s*i end; _____
4) s:=1; i:=6;

while i>=3 do begin s:=s+i; i:=i–1 end; _____
а) 18 б) 2.2 в) 0 г) 5 д) 10 е) 48

4) *Вказати значення змінної S, якого вона набуде після виконання таких операторів:*

1) s:=4; i:=1;
repeat s:=s+1/i; i:= i–1 until i<=1; _____
2) s:=2; i:=1;
repeat s:=s+1/(i+1); i:=i+3 until i>=5; _____
3) s:=1; i:=1;
repeat i:=i+1; s:=s+i until i>3; _____
4) s:=2; i:=5;
repeat i:=i–1; s:=s+1/i until i<=4; _____
а) 2.25 б) 2.7 в) 7 г) 10 д)5

5) *Вказати значення змінної S, якого вона набуде після виконання таких операторів:*

1) s:=0; for i:=10 downto 6 do s:=s+1; _____
2) s:=0; for i:=8 to 3 do s:=s+1; _____
3) s:=1; for i:=2 downto 10 do s:=s+1; _____
4) s:=1; for i:=5 to 9 do s:=s+1; _____
а) 6 б) 1 в) 5 г) 2 д) 0 е) 40

6) ***Практичне завдання.*** *Дано натуральне число N:*

- знайти добуток його цифр;
- чи містить дане число цифру A (A вводиться з клавіатури)?

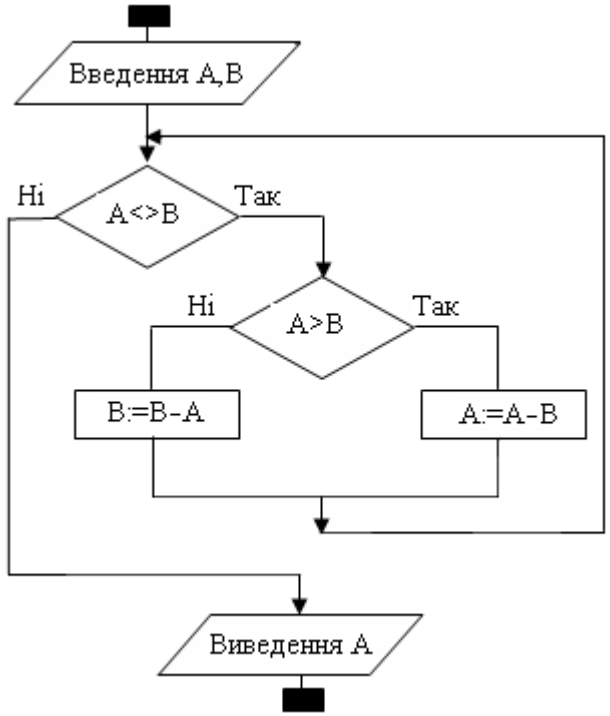
1.12. Алгоритм Евкліда

Для обчислення найбільшого спільного дільника (НСД) двох чисел уже більше 20-ти століть відомий алгоритм, запропонований давньогрецьким математиком Евклідом – ***алгоритм Евкліда.*** Визначити НСД можна двома способами, які відрізняються швидкістю виконання.

1-й спосіб. Порівнюються два числа і більше з них замінюється різницею цих чисел. Це повторюють до того часу, поки числа не стануть рівними. Отриманий результат і є шуканим НСД двох чисел. Фрагмент програми для цього способу може мати вигляд:

```
Write ('Введіть число A= ');
Readln (A);
Write ('Введіть число B= ');
Readln (B);
```

```
 While A<>B do
   If A>B Then A:=A-B   Else B:=B-A;
Writeln ('НСД=',A);
```



2-й спосіб. Для обчислення НСД двох чисел більше з них замінюється остачею від ділення більшого числа на менше до того часу, доки одне з чисел не стане рівним 0. Тоді інше число є НСД двох чисел. З використанням циклу `Repeat` фрагмент програми матиме такий вигляд:

```
Write ('Введіть число A= ');
Readln (A);
Write ('Введіть число B= ');
Readln (B);
 Repeat
   If A>B Then A:=A mod B Else B:=B mod A
 Until (A=0) or (B=0);
Writeln ('НСД=', A+B);
```


Цикли `While` і `Repeat` в даних фрагментах можна замінювати один одним, але при цьому потрібно звертати особливу увагу на складання умов виконання та зупинки циклів.

Два числа, найбільший спільний дільник яких дорівнює одиниці, називаються ***взаємнопростими***.

Питання для самоконтролю:

1. *У чому полягає метод знаходження НСД двох цілих чисел з використанням різниць?*
2. *У чому полягає метод знаходження НСД двох цілих чисел з використанням остач від ділення?*
3. *Проаналізуйте знаходження НСД для A=18, B=3. Яким способом НСД знаходиться швидше?*
4. *Які числа називаються взаємнопростими? Наведіть приклади.*

Задачі

1) *Знайти НСД трьох чисел.*
   Примітка. НСД **(**a, b, c**)**=НСД(НСД(a, b), c).
2) *Перевірити, чи є два даних числа взаємнопростими.*
3) *Знайти найменше спільне кратне (НСК) чисел n і m, використовуючи співвідношення* $НСК(n,m)=\frac{nm}{НСД(n,m)}$.

1.13. Вкладені цикли

Якщо при розв'язуванні задач, потрібно, щоб змінювалась не одна змінна, а кілька, то один цикл розміщується всередині іншого. Такі конструкції називають ***вкладеними циклами***.

Розглянемо роботу вкладених циклів на прикладах.

Приклад №1

Написати програму, яка друкує таблицю множення за схемою Піфагора для заданого проміжку цілих чисел. Наприклад:

	1	**2**	**3**	**4**	**5**	**6**	**7**
1	1	2	3	4	5	6	7
2	2	4	6	8	10	12	14
3	3	6	9	12	15	18	21
4	4	8	12	16	20	24	28

Розв'язування

```
Program Tabliczka;
Var i,j,k,w: integer;
```

```
Begin
  Write('Введіть кількість стовпчиків: ');
  ReadLn(k);
  Write('Введіть кількість рядків: ');
  ReadLn(w);
  Write(' *');
  For j:=1 to k do Write(j:4);
  WriteLn;
  For i:=1 to w do
   Begin
     Write(i:2);
     For j:=1 to k do Write(i*j:4);
     WriteLn
   End
End.
```

Приклад №2

Якщо додати всі цифри якогось числа, потім усі цифри знайденої суми і далі повторювати цей процес, то ми одержимо одноцифрове число (цифру), яке називають ***цифровим коренем*** даного числа. Наприклад, цифровий корінь числа 34 697 дорівнює 2 (3+4+6+7=29; 2+9=11; 1+1=2). Складемо програму для обчислення цифрового кореня натурального числа.

Розв'язування

Програма, що обчислює цифровий корінь даного числа, може виглядати так:

```
Program Example;
Var n,k,s:integer;
Begin
   Write('Введіть число:');
   Readln(n);
   s:=n;
   While s>9 Do {поки число багатоцифрове…}
   Begin
   k:=s; s:=0;  {обчислити суму його цифр}
    Repeat
      S:=s+k mod 10;
      K:=k div 10
   Until k=0
   End;
```



```
   Writeln('Цифровий корінь числа ',n,' = ',s);
End.
```

Приклад №3

Знайти всі натуральні числа a, b і c з інтервалу від 1 до 20, для яких виконується рівність: $a^2 + b^2 = c^2$.

Розв'язування

```
Program Example;
Var
   a,b,c:integer;
Begin
   for a:=1 to 20 do
      for b:=1 to 20 do
         for c:=1 to 20 do
            if sqr(a)+sqr(b)=sqr(c) then
               Writeln('a=',a,' b=',b,' c=',c)
End.
```

Питання для самоконтролю:

1. *Для чого в програмі з прикладу №1 оператор Write(' *')?*
2. *Проаналізуйте роботу програми з прикладу №2 для n=5, n=56.*
3. *Замініть в програмі з прикладу №2 цикл Repeat на цикл While.*
4. *Проаналізуйте, скільки разів за весь час роботи програми з прикладу №3 виконуватиметься кожен цикл.*
5. *Проаналізуйте, скільки разів за весь час роботи програми з прикладу №3 виконуватиметься перевірка умови в умовному операторі.*

Задачі

1) *Що буде надруковано після виконання такого фрагменту програми при n=6?*

```
a:=1;b:=1;
for i:=0 to n do
      Begin
         For j:=1 To b Do write('*');
         Writeln;
         c:=a+b; a:=b; b:=c
      End;
```

Розв'язок якої задачі реалізовано в цьому фрагменті?

2) *Знайти всі такі трійки натуральних чисел x, y і z з інтервалу від 1 до 20, для яких виконується рівність* $x^2-y^2=z^2$.

3) *Знайти цілі числа із проміжку від 1 до 200, у яких рівно 5 дільників.*

1.14. Дійсний тип даних

До **дійсного (real)** типу належить підмножина дійсних чисел, які можуть бути подані у форматі з плаваючою комою і з фіксованою кількістю цифр. Дійсні числа в АЛГО мають порядок від E+320 до E–320.

Число з фіксованою кількістю цифр зображується десятковим дробом (дробова частина може бути й нульовою). Дробова частина відокремлюється від цілої за допомогою крапки.

Наприклад: 127.3; 25.0; – 45.004; 0.77.

Число з плаваючою комою має вигляд *m*E*p*, де *m* – мантиса, яка може бути цілим або дійсним числом з фіксованою крапкою, а p – порядок числа, який має тип `integer`. Як мантиса, так і порядок можуть містити знаки "+" або "–". Наприклад:

Математичний запис	Запис із плаваючою комою
0,000009	9E–6
$0{,}62*10^4$	0.62E+4
$-10{,}8*10^{12}$	–1.08E13
$20*10^{-3}$	2E–2

До даних дійсного типу застосовуються математичні операції: «+», «-», «*», «/». Результати цих операцій теж матимуть тип `real`.

Є багато стандартних функцій для роботи з дійсними числами. Перерахуємо ті, що найчастіше використовуються:

`Sqrt(x)` – квадратний корінь з x (x має бути не від'ємним, цілим або дійсним числом);

`Trunc(x)` – ціла частина від x;

`Round(x)` – округлення до найближчого цілого числа.

До даних дійсного типу застосовують і відомі вам функції **`Abs(x)`** та **`Sqr(x)`**.

Цікавою є функція, яка може вибирати випадкові значення із заданого діапазону. Вона має назву `Random` або **Випадкове** і може використовуватись з аргументом або без нього:

`Random`
`Random (x)`

Якщо функцію **Випадкове** вживати без аргумента, то результатом є **дійсне** випадкове число в діапазоні **[0, 1]**. У варіанті з фактичним параметром, яким може бути вираз **дійсного** або **цілого** типу, результат має такий самий тип, як аргумент, а його значення належить до діапазону **[0,x]**. Зрозуміло, що комп'ютер випадково нічого не робить. "Випадкові" числа генеруються за спеціальним алгоритмом на основі деякого цілого числа. Це число утворюється автоматично з показників системного годинника у момент запуску програми на виконання, тому під час кожного виконання програми генерується інша послідовність випадкових чисел.

При запису команди присвоювання потрібно пам'ятати, що змінній дійсного типу можна надати значення виразу цілого типу, але не навпаки. Якщо у виразі використовуються дані цілого і дійсного типів, то результат матиме дійсний тип.

Приклад:
```
Var a,b:integer;
    c:real;
```
допустимі такі записи команди присвоєння:
```
c:=Sqrt(a);
c:=2*b;
c:=c+a;
c:=a/b.
```

Виводити дані дійсного типу можна за заданим формату й без нього. Якщо під час виведення даних дійсного типу не зазначений формат, то число виводиться з плаваючою крапкою – мантиса і порядок. На число відводиться 17 позицій. При цьому в цілій частині мантиси присутня тільки одна значуща цифра. Змінити стандартну форму виведення можна, використовуючи формат: **`Write(x:m:n)`**, де x – значення дійсного типу, що виводиться; m – загальна кількість позицій для виведення числа (включно зі знаком числа, цілою частиною, крапкою та дробовою частиною); n – кількість позицій для виведення дробової частини.

Приклад

Надрукувати таблицю значень функції $y=x^2$ на відрізку [0,5] із кроком 0.5.

Розв'язування

Для перебору всіх значень з відрізка в даному випадку безпосередньо використати цикл із параметром неможливо,

оскільки крок зміни значення параметра – дійсне число. Тому застосуємо цикл **While**.

```
Program Example;
Var  i:real;
Begin
  i:=0;
  While i<=5 do
   Begin
   Writeln(i:2:1,' ',sqr(i):4:3);
   i:=i+0.5
   End
End.
```

Питання для самоконтролю:

1. *Що таке дійсний тип даних та для чого він використовується?*
2. *Як задаються значення змінних з плаваючою комою?*
3. *Як вказують формат виведення дійсних чисел?*
4. *Які стандартні функції використовуються для роботи з дійсними числами?*
5. *Вкажіть неправильно записані оператори присвоєння, якщо:*

```
var a, b:Real; c:Integer;
```

а) a := b * c; б) c := Sqrt(c);
в) a := c mod a; г) c := Abs(a + b);
д) a := c mod a; е) c := Trunc(a + b).

6. *Обчисліть значення виразів, якщо:*

```
var a, b:Real; c:Integer;
begin
    …
    a := 3.15; b := –6; c := 15;
    …
end.
```

а) c := c + 1; б) a := a – c /100;
в) b := b + c div 2 – 1; г) b := Trunc(a) + c;
д) a := Trunc(a) + Abs(b) + c;

Задачі

1) *Дано дійсні додатні числа* a, b, c, x, y. *З'ясувати, чи пройде цеглина з ребрами* a, b, c *у прямокутній отвір зі сторонами* x, y. *Просувати цеглину дозволяється тільки так, щоб кожне з її ребер було перпендикулярне або паралельне кожній зі сторін отвору.*

2) *Задаються коефіцієнти квадратного рівняння* $ax^2+bx+c=0$. *Скласти програму для обчислення його коренів.*
3) *Дано три дійсні числа* a, b, c. *Знайти середнє арифметичне цих чисел.*

1.15. Символьний тип даних

Символьний тип даних, як і цілий та логічний, відноситься до порядкових типів. Дані символьного типу описуються за допомогою ідентифікатора **Char.**

Значенням змінної символьного типу може бути будь-який символ – букви, цифри, розділові знаки та спеціальні символи. Кожному символу відповідає унікальний числовий код від 0 до 255.

Найпоширенішою міжнародною системою кодування символів є система ASCII – Американський стандартний код для обміну інформацією. Символи з кодами від 0 до 127 утворюють основну таблицю кодів ASCII, серед яких коди від 0 до 31 – службові. Основна таблиця кодів однакова на всіх IBM-сумісних комп'ютерах. Символи з кодами від 128 до 255 утворюють так звану національну кодову таблицю. Саме в ній розташовуються, наприклад, українські літери.

Таблиця кодів ASCII:

32		33	!	34	«	35	#	36	$
37	%	38	&	39	'	40	(	41	)
42	*	43	+	44	,	45	–	46	.
47	/	48	0	49	1	50	2	51	3
52	4	53	5	54	6	55	7	56	8
57	9	58	:	59	;	60	<	61	=
62	>	63	?	64	@	65	A	66	B
67	C	68	D	69	E	70	F	71	G
72	H	73	I	74	J	75	K	76	L
77	M	78	N	79	O	80	P	81	Q
82	R	83	S	84	T	85	U	86	V
87	W	88	X	89	Y	90	Z	91	[
92	\	93	]	94	^	95	_	96	`
97	a	98	b	99	c	100	d	101	e
102	f	103	g	104	H	105	i	106	j
107	k	108	l	109	M	110	n	111	o
112	p	113	q	114	R	115	s	116	t
117	u	118	v	119	W	120	x	121	y

122	z	123	{	124	|	125	}	126	~
127	□	128	Ъ	129	Ѓ	130	‚	131	ѓ
132	„	133	…	134	†	135	‡	136	€
137	‰	138	Љ	139	‹	140	Њ	141	Ќ
142	Ћ	143	Џ	144	Ъ	145	‘	146	’
147	“	148	”	149	•	150	–	151	—
152	□	153	™	154	Љ	155	›	156	њ
157	ќ	158	ћ	159	Џ	160		161	Ў
162	ў	163	J	164	¤	165	Ґ	166	¦
167	§	168	Ё	169	©	170	Є	171	«
172	¬	173	-	174	®	175	Ї	176	°
177	±	178	I	179	I	180	ґ	181	µ
182	¶	183	·	184	ё	185	№	186	є
187	«	188	j	189	S	190	s	191	ї
192	А	193	Б	194	В	195	Г	196	Д
197	Е	198	Ж	199	З	200	И	201	Й
202	К	203	Л	204	М	205	Н	206	О
207	П	208	Р	209	С	210	Т	211	У
212	Ф	213	Х	214	Ц	215	Ч	216	Ш
217	Щ	218	Ъ	219	Ы	220	Ь	221	Э
222	Ю	223	Я	224	А	225	б	226	в
227	г	228	д	229	е	230	ж	231	з
232	и	233	й	234	к	235	л	236	м
237	н	238	о	239	п	240	р	241	с
242	т	243	у	244	ф	245	х	246	ц
247	ч	248	ш	249	щ	250	ъ	251	ы
252	ь	253	э	254	ю	255	я		

До символьних даних застосовують операції порівняння. Операція порівняння здійснюється таким способом: з двох символів «менший» той, який зустрічається у таблиці ASCII раніше.

Значення для змінних типу `Char` задаються в апострофах: `ch:='*'; a:='3'; letter:='G'.` Функція **`Chr(x)`** повертає символ за його кодом (номером в таблиці кодів). Зворотну операцію знаходження номера символу ch виконують за допомогою функції **`Ord(ch).`**

Приклад №1

Написати програму для виведення «трикутника» символів:

A
AB
ABC
…
AB… YZ.

Розв’язування

«Трикутник» будується за таким правилом: послідовно виводяться частини латинського алфавіту, що складаються з одного символу, з 2-х символів, з 3-х символів – і так доти, доки не буде виведений весь алфавіт. Кількість таких послідовностей дорівнює кількості літер алфавіту. Оскільки символьний тип даних є порядковим типом, то змінну цього типу можна використовувати як параметр циклу:

```
Program Example;
Var i,j:Char;
Begin
   For i:='A' to 'Z' do
   Begin
      For j:='A' to i do Write (j);
      Writeln
   End
End.
```

Приклад №2

Написати програму для підрахунку кількість цифр, які входять у заданий текст, що закінчується крапкою. Текст в даному випадку – послідовність символів, введення кожного з яких завершується натисканням клавіші Enter.

Розв’язування

Будемо вводити символи доти, доки значення чергового символу не співпаде зі значенням ‘.’. Аналізуючи кожне значення, будемо збільшувати лічильник, якщо символ є цифрою:

```
Program Example;
Var ch:Char;
   k:Integer;
Begin
 Read (ch);
 k:=0;
 While ch<>'.' do
   Begin
    If (ch>='0') and (ch<='9') then k:=k+1;
```

```
    Read(ch);
  End;
 Writeln ('Кількість цифр: ',k);
End.
```

Питання для самоконтролю:

1. *Що таке символьний тип даних та для чого він використовується?*
2. *Як задаються значення змінних символьного типу?*
3. *Що таке таблиця кодів ASCII?*
4. *Які стандартні функції використовуються для роботи з символьними величинами?*

Задачі

1) *Модифікувати програму прикладу №2 так, щоб за її допомогою можна було визначити, чи є текст правильним записом цілого числа.*
2) *Напишіть програму, при виконанні якої після введення символу виводиться на екран його код.*
3) *Напишіть програму для виведення на екран символів таблиці ASCII з кодами від 32 до 255.*
4) *Скласти програму для визначення того, яка літера частіше зустрічається у введеному тексті: ‘а’ чи ‘б’?*
5) **Дано послідовність символів, що містить n доданків і має такий вигляд: d1±d2±…±dn (d1, d2 і т.д. – цифри, ± – знак «+» або «–», n>1). Обчислити значення виразу.*

1.16. Рядковий тип даних

Розв’язуючи задачі, нам доводиться виконувати не тільки математичні обчислення, а й опрацьовувати текстову інформацію: знаходити потрібне слово, виконувати електронний переклад, шифрувати або розшифровувати текст і т.д. Для роботи з текстами в Паскалі застосовується структурований тип даних `String` (рядок). *Рядком* називається послідовність певної довжини, що складається з символів.

Опис типу:

```
Var  Str:string;
```

Найбільша довжина рядка не перевищує 255 символів і може бути вказана в його описі. Наприклад, рядкові змінні з найбільшою довжиною 10 символів описують так:


```
Var  S,T:string[10];
```

Якщо найбільша довжина в описі не вказана, то вона становить 255 символів. Справжня довжина рядка залежить від присвоєного значення і не перевищує вказаної в описі. Для описаних вище змінних можливі такі присвоєння:

```
Str:='мир'; {справжня довжина 3 символи}
S:='я і ти';{справжня довжина 6 символів}
T:='';{справжня довжина 0 символів}
```

Як бачите, рядкові значення в програмі обмежуються апострофами. Якщо ж рядок теж містить апостроф, то в тексті програми цей символ слід подвоїти:

```
S:='Сім''я'; {присвоєне значення «Сім'я»}
```

Змінні типу `String` виводяться на екран за допомогою стандартних процедур `Write` і `Writeln` і вводяться за допомогою стандартних процедур `Readln` і `Read`. Тобто вводяться і виводяться не поелементно, як символи, а повністю.

Операції з рядками

У Паскалі є два основних способи опрацювання змінних типу **`String`**. Перший спосіб передбачає опрацювання всього рядка як єдиного цілого, тобто єдиного об’єкта. Також можна розглядати рядок як складний об’єкт, що складається з окремих символів, тобто елементів типу `Char`, що під час опрацювання доступні кожний окремо. До кожного символу можна звернутися за його номером: `Str[i]` – це звернення до *i*-го символу рядка `Str`. Нумерація символів починається з `1`.

1. Склеювання

Під склеюванням розуміється послідовне об’єднання кількох рядків в один.

Приклад

```
Var Str1, Str2, Str3:string;
begin
Str1:='Я люблю';
Str2:='інформатику.';
Str3:=Str1+ ' ' +Str2;
...
```

Рядок `Str3` набуває значення «Я люблю інформатику.».

2. Порівняння

Паскаль дає змогу виконувати операції порівняння двох рядків. Порівняння відбувається посимвольно зліва направо: порівнюються коди відповідних символів, доки не порушиться рівність або не закінчиться один з рядків (чи обидва разом). При цьому відразу робиться висновок про знак нерівності. Два рядки називаються *рівними*, якщо вони рівні за довжиною і збігаються посимвольно.

Приклад

```
'Balkon'<'balkon' (Ord('B')<Ord('b'));
```

'Кіт'='Кіт' (рівні за довжиною і збігаються посимвольно).

Стандартні процедури і функції

АЛГО надає у розпорядження цілу низку стандартних функцій і процедур мови Паскаль, призначених для опрацювання рядків. Розглянемо найважливіші з них. Назви функцій подані двома мовами.

1. Вилучення

Для вилучення з рядка фрагмента використовується процедура **Вилучити** Delete(Str,n,m), що вирізає з рядка Str m символів, починаючи з n–го. Таким чином, після виклику процедури Delete рядок змінюється.

Приклад

```
Str1:='ABCDEFGH';
Delete (Str1,3,4);
```

Після виконання цих операторів із рядка будуть вилучені чотири символи (починаючи з третього), тобто рядок стане таким: Str1='ABGH'.

2. Вставка

Для вставки одного рядка в інший використовується процедура **Вставити** Insert (Str1,Str2,n), що вставляє рядок Str1 у рядок Str2, починаючи з n–го символу. При цьому перший рядок не змінюється , а другий одержує нове значення.

Приклад

```
Str1:='ABCDEFGH';
Str2:='abcdefgh';
Insert(Str1, Str2, 3);
```


У результаті виконання даної процедури рядок стане таким: Str2= 'abABCDEFGHcdefgh'. Таким самим буде результат виконання такої послідовності операторів:

```
Str2:='abcdefgh';
Insert ('ABCDEFGH',Str2,3);
```

3. Копіювання

Функція **Фрагмент** Copy (Str,n,m) копіює m символів рядка str, починаючи з n-го символу. При цьому рядок Str не змінюється.

Приклад

```
Str1:='ABCDEFGH';
Str2:='abcdefgh';
Str3:=Copy(Str1,4,3);
Writeln(Str3);
Writeln(copy(Str2,4,3));
```

Значення змінної Str3='DEF'. А на екран будуть виведені такі рядки:

```
DEF
def
```

4. Довжина рядка

Під довжиною рядка розуміють фактичну (а не найбільшу можливу) кількість символів у рядку. Її значення можна отримати за допомогою функції **Довжина** Length (Str). Результат її виклику – ціле число, що дорівнює кількості символів рядка Str.

Приклад

```
Str:='Я люблю інформатику.';
K:=Length(Str);
```

У результаті значення змінної K буде дорівнювати 20.

5. Пошук підрядка

Функція **Знайти** Pos (Str1,Str2) визначає позицію підрядка в рядку. Її результат – ціле число. Воно визначає номер елемента, з якого починається перше входження підрядка Str1 у рядок Str2. Якщо Str1 не входить у Str2, то значення функції дорівнює 0.

Приклад

```
K:=Pos('ка', 'солодка карамелька')
```

У цьому випадку значення K дорівнює 6, тому що функція Pos повертає номер елемента, починаючи з якого підрядок зустрічається перший раз.

6. Числа і рядки

Для роботи з числами й рядками застосовують дві процедури перетворення типів.

Текст `Str(N,Str1)` – переводить числове значення `N` у рядкове і присвоює результат рядку `Str1`, причому можна переводити як цілі числа, так і дійсні.

Приклад

`Str(1234, Str1)` – після виконання `Str1='1234';`

`Str(452.567, Str1)` – переводимо дійсне число, результат `Str1='452.567';`

Друга процедура виконує обернену дію.

Значення Val (Str, N, K) – переводить рядкове значення у числове. Якщо даний рядок є записом числа (цілого або дійсного), то значення K стане рівне 0, а N – шуканому числу. Якщо ж повністю перетворити рядок в число не вдалось, то K буде рівним номеру помилкового символу в рядку, а N – числу, записаному символами перед помилковим.

Наведемо програму, яка демонструє різні варіанти виклику процедури **Значення**:

```
Program ValDemo;
Var x, k:integer;
   y:real;
Begin
 Val('1234', x, k);
 Write(k); If k=0 then WriteLn(' x=',x) else WriteLn;
 Val('abc', x, k);
 Write(k); If k=0 then WriteLn(' x=',x) else WriteLn;
 Val('-1.3E-3', y, k);
 Write(k); If k=0 then WriteLn(' y=',y) else WriteLn;
 Val('-1.3E+0', y, k);
 Write(k);  If  k=0  then  WriteLn('y=',y:0:1)  else
WriteLn;
end.
```

Під час виконання програми виводиться така інформація:

```
 0 x=1234
 1
 0 y=-1.300000E-003
 0 y=-1.3
```


Приклад №1 Скільки разів у заданому рядку зустрічаються символ ‘a’?

Розв’язування

Складемо програму, в якій будемо опрацьовувати рядок. Результат виконання – ціле число. Переглядаємо всі символи рядка (їх кількість дорівнює довжині рядка). Якщо символ дорівнює ‘a’, то збільшуємо лічильник.

```
Program Example;
Var st:String;
    i,k:integer;
Begin
 Write('Введіть рядок- ');
 Readln (st);
 k:=0;
 For i:=1 to length (st) do
    If st[i]='a' then k:=k+1;
 Writeln('Кількість букв ''a'' - ',k)
End.
```

Приклад №2 Замінити всі включення підрядка ‘del’ на ‘insert’.

Розв’язування

Якщо такий підрядок є, то необхідно знайти номер першого символу підрядка ‘del’, вилучити його і вставити ‘insert’. Так повторювати до того часу, поки `Pos('del',st)<>0`.

```
Program Example;
Var st:String;
    k:integer;
Begin
 Write('Введіть рядок - ');
 Readln (st);
 While Pos('del',st)<>0 do
  Begin
    K:=Pos('del',st);
    Delete(st,k,length('del'));
    Insert('insert',st,k)
  End;
 Writeln('Змінений рядок - ',st)
End.
```

Питання для самоконтролю:

1. *Що таке рядковий тип даних та для чого він використовується?*
2. *Які стандартні функції використовуються для роботи з рядковими величинами?*
3. *Вкажіть, яким вказівкам присвоєння з лівого стовпчика відповідають значення L з правого.*

1) L: = Length ('Pascal');	____	а) 9
2) L: = Length ('Фірма IBM');	____	б) 0
3) L: = Length ('');	____	в) 8
4) L: = Length ('АЛГО');	____	г) 6
5) L: = Length ('Програма');	____	д) 4

4. *Якими будуть значення змінних K1 і K2 після виконання заданого фрагменту програми?*

```
Str1:= 'CDE';   Str2:='ABCDEFGH';
K1:= Pos(Str1, Str2);
K2:= Pos(Str2, Str1);
```

5. *Дано: R:='Константинополь';*

Вкажіть, які значення з правого стовпчика отримає змінна R після виконання вказівки з лівого:

1) Delete(R,1,5);	_____	а) 'антинополь'
2) Delete(R,5,8);	_____	б) 'Констанополь'
3) Delete(R,8,3);	_____	в) 'Консоль '

6. *Дано: Const R ='Константинополь';*

Вкажіть, яким вказівкам присвоєння з лівого стовпчика відповідають значення S з правого стовпчика:

1) S:=Copy(R, 4, 4);	____	а) 'тин'
2) S:=Copy(R, 8, 3);	____	б) 'анти'
3) S:=Copy(R, 6, 4);	____	в) 'стан'

Задачі

1) *Підрахуйте, скільки разів у даному рядку зустрічається введений з клавіатури символ.*
2) *Замінити всі символи Ch1 у рядку на Ch2 (значення змінних Ch1 і Ch2 вводяться з клавіатури).*
3) *Замінити в рядку всі входження підрядка Str1 на підрядок Str2 (Str1 і Str2 вводяться з клавіатури).*
4) *Дано рядок, що складається з кількох слів, між словами один пропуск, у кінці рядка – крапка. Підрахувати кількість слів у рядку.*

5) **Знайдіть довжину найдовшого та найкоротшого слів у заданому рядку. Слова відокремлені одиничними пропусками.*

1.17. Підготовка до оцінювання з теми «Типи даних»

1) *Вкажіть правильні твердження: «Тип даних…*

а) вказує на кількість значень у певній числовій множині»;

б) визначає скінченну множину числових значень»;

в) визначає скінчену множину допустимих для нього значень та операції, які можна виконувати над такими даними”.

2) *Які значення належать до типу Real:*

а) 2	г) 2.5E+2
б) 1.7	д) –5.37
в) ’37.5’	е) Sqr(3).

Вкажіть значення параметра ch після виконання оператора For:

3) *For ch:= ‘a’ to ‘z’ do …*

а) ‘a’; б) ‘z’; в) ‘b’; г) 64; д) не визначене.

4) *For ch := ‘d’ to ‘a’ do …*

а) ‘d’; б) ‘a’; в) ‘c’; г) 64; д) не визначене.

5) *Виправте помилки в наведених фрагментах програм:*

```
i := 1;
for ch := 'a'…'z' do
begin
  writeln(i, ' ', ch);
   i:= i + 1;
end;
```

```
for ch:= 'a' to 'z'
begin
    write(ch, ' ', Ord(ch));
end;
```

6) *Яким буде значення змінної K після виконання такого фрагменту програми:*

K: = Pos ('in', 'Interesting');?

7) *Яким буде значення змінної S після виконання кожного з фрагментів програми:*

a) S:='computer';	b) S:='program';	d) P:= 'Internet';
Delete (S, 3, 4);	Insert ('no', S, 3);	S:=Copy (P, 4, 3);

8) *Заповніть пропуски так, щоб значення змінної S після виконання заданого фрагменту програми було рівним 'пост':*

```
S: = 'постанова';
Delete (…,  ..., ...);
```

9) *Заповніть пропуски так, щоб значення змінної S після виконання заданого фрагменту програми було рівним 'форматування':*

```
S:= 'формування';
Insert (..., ..., ...);
```

10) *Заповніть пропуски так, щоб значення змінної S після виконання заданого фрагменту програми було рівним 'ритм':*

```
P:= 'алгоритмізація';
S:= Copy (..., ..., ...);
```

Практичні завдання

11) *Знайти площу кільця, якщо відомі довжини його радіусів.*

12) *Дано два рядки. Вивести символи, які зустрічаються як в першому, так і в другому рядках.*

2. Елементи структуризації програми

2.1. Підпрограми-процедури

Часто у програмах зустрічаються повторювані або схожі фрагменти. В мові програмування Паскаль існує можливість оформляти такі фрагменти спеціальним чином – виділяти їх у *підпрограми.* Підпрограмі дається ім'я, за яким можна звертатися до неї (викликати підпрограму). Використання підпрограм не тільки покращує структуру та зовнішній вигляд програми, але й зменшує можливість виникнення помилок і полегшує відлагодження.

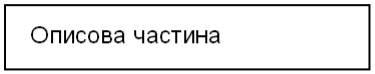
Є два види підпрограм – процедури і функції. Їх структура дуже схожа на структуру основної програми.

У програмі процедури і функції описуються після розділу опису змінних програми, але до початку розділу операторів, тобто до `Begin,` що починає цей розділ.

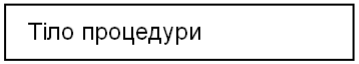
Опис процедури

Опис процедури починається із заголовка, що є обов'язковим. Заголовок складається зі службового слова `Procedure,` за яким вказують ім'я процедури й, можливо, список формальних параметрів у круглих дужках. Наприкінці заголовка ставиться крапка з комою. Після заголовка можуть розміщуватись ті ж розділи, що й у програмі: описи змінних, процедур функцій тощо. Загальний вигляд опису процедури (у квадратні дужки взято

```
Procedure <ім'я> [(Список формальних параметрів)];
```




```
Begin
```




```
End;
```

частину, якої може не бути):

Під час виклику процедури її формальні параметри замінюються відповідними фактичними. *Фактичні параметри* – це параметри, що передаються процедурі під час її виклику. *Кількість і типи формальних та фактичних параметрів повинні повністю збігатися.*

Формальні параметри описуються у заголовку процедури й визначають тип і місце підстановки фактичних параметрів. Формальні параметри діляться на два види: *параметри-змінні та параметри-значення*.

Параметри-змінні відрізняються тим, що перед ними стоїть службове слово Var. Вони використовуються тоді, коли необхідно, щоб зміни в тілі процедури значень формальних параметрів призводили до змін відповідних фактичних параметрів.

Параметри-значення відрізняються тим, що перед ними слово Var не ставиться. У середині процедури можна робити будь-які дії з параметрами-значеннями, але всі зміни ніяк не відбиваються на значеннях відповідних фактичних параметрів, тобто якими вони були до виклику процедури, такими й залишаться після завершення її роботи.

Всі змінні програми діляться на *глобальні й локальні*.

Глобальні змінні оголошуються у розділі описів головної програми.

Локальні змінні оголошуються у процедурах і функціях. Таким чином, локальні змінні «живуть» тільки під час роботи підпрограми.

Ознакою гарного стилю програмування є уникнення невиправданого використання глобальних змінних.

Приклад №1

Скласти програму для обчислення a^n: цілі числа a і n ($n \geq 0$) вводяться з клавіатури.

Розв'язування

Складемо процедуру для обчислення степеня цілого числа:

```
Procedure  Step (x,y:Integer; Var st:integer);
 Var i:Integer;        {описова частина}
Begin                  {тіло процедури}
     St:=1;
     For i:=1 To y Do st:=st*x
End;
```

Перший рядок – це заголовок процедури, що починається зі слова Procedure. Процедура названа ім'ям Step. У дужках записаний список формальних параметрів, тобто перелічені змінні і вказаний їх тип.

Ми використовуємо три параметри: перший – основа степеня, тобто число, яке потрібно піднести до степеня, другий – показник степеня, третій – змінна, в яку буде переданий результат.

Перші два формальні параметри – параметри-значення, третій – параметр-змінна, тому перед ним написане слово Var. Усі вони описані як цілі.

Після заголовка процедури йдуть розділи описів. У нашому прикладі описується одна змінна і (параметр циклу). Далі йде тіло процедури. Воно починається зі службового слова Begin і закінчується службовим словом End, після якого стоїть крапка з комою. У тілі процедури обчислюється степінь числа x за допомогою циклу For.

Вся програма для розв'язування задачі може мати такий вигляд (процедура виділена коментарями для кращого сприйняття):

```
Program Example;
Var  a,n,s:Integer;
{---------------------------------}
Procedure Step(x,y:integer; Var st:integer);
Var i:integer;
Begin
   st:=1;
   For i:=1 To y Do st:=st*x;
End;
{---------------------------------}
Begin
   Write ('Введіть два числа - основу та по-
казник степеня:');
   Readln (a,n);
   Step (a,n,s);     {звернення до процедури}
   Writeln ('Результат = ', s)
End.
```

Процедура викликається як оператор, що складається з імені процедури. У круглих дужках записуються фактичні параметри. У нашому прикладі формальні параметри x, y і st набирають значення фактичних параметрів a, n і s відповідно. Після завершення роботи процедури змінні a і n збережуть ті самі значення, що мали під час виклику, а s одержить нове значення.

Приклад №2

Дано дві цілі змінні. Поміняти місцями їх значення.

Розв'язування

Поміняти місцями значення двох змінних можна двома способами – через проміжну змінну або без неї. Напишемо процедуру, що реалізує перший спосіб.

```
Procedure Swap (Var x, y:integer);
     Var z:integer;
Begin
     z:=x; x:=y; y:=z
End;
```

Процедура називається Swap. У неї є два формальних параметри, що є параметрами-змінними, тому що необхідно поміняти значення змінних і запам'ятати зміни. Ці параметри є результатами виконання процедури. У процедурі описана змінна z, що використовується як проміжна. Вся програма має вигляд:

```
Program Example;
Var a, b: integer;
{--------------------------------}
 Procedure Swap (Var x, y:integer);
 Var z: integer;
   Begin
      z:=x; x:=y; y:=z
   End;
{--------------------------------}
Begin
   Write ('Введіть значення змінних a і b:');
   Readln (a, b);
   Swap(a, b);          {звернення до процедури}
   Writeln ('a=',a,' b=',b) {виведення нових значень}
End.
```

Питання для самоконтролю:

1. *Що таке підпрограма? Для чого використовуються підпрограми? Вкажіть переваги використання підпрограм.*
2. *Що таке формальні та фактичні параметри?*
3. *Які змінні називаються глобальними, а які локальними? Наведіть приклади.*
4. *Який загальний вигляд оформлення процедур?*
5. *Як виглядає виклик процедури в головній програмі?*

Задачі

1) *Використовуючи процедуру для обчислення степеня числа, знайти значення виразу:*

$$y=a_4x^4+a_3x^3+a_2x^2+a_1x+a_0.$$

Коефіцієнти a_4, a_3, a_2, a_1, a_0 і значення x вводяться з клавіатури.

2) *Використовуючи процедуру, розглянуту в прикладі 2, впорядкувати за зростанням значення трьох змінних a, b, c.*
3) *Дано натуральне число. Написати програму з використанням процедури, яка друкує всі його дільники й підраховує їх кількість.*

2.2. Підпрограми-функції

Розглянемо другий вид підпрограм – функції. Відмінність від процедур полягає в тому, що результатом виконання функції є єдине значення. Це обчислюване значення присвоюється імені функції.

Опис функції

Загальний вигляд опису функції такий:

```
Function <ім'я>(Список формальних параметрів):тип результату;
```

Описова частина

```
Begin
```

Тіло функції, у якому обов'язково повинно бути присвоювання
Ім'я функції: = значення;

```
End;
```

Як бачите, заголовок функції складається зі слова Function, за яким вказується ім'я функції, потім у круглих дужках записується список формальних параметрів. Далі ставиться двокрапка і вказується тип результату функції.

У тілі функції обов'язково повинен бути хоча б один оператор присвоєння, у лівій частині якого стоїть ім'я функції, а в правій – її значення. Інакше значення функції не буде визначено.

Звертання до функції в головній програмі здійснюється у виразі. При цьому вказують її ім'я та список фактичних параметрів.

Приклад №1

Написати функцію для підрахунку кількості цифр натурального числа. Використовуючи її, визначити, у якому з двох даних чисел більше цифр.

Розв'язування

Ми вже розв'язували схожу задачу, але в ній не вимагалося написати функцію.

```
Function Quantity(x:integer):integer;
     Var k:integer;
Begin
     k:=0;
     While x<>0 Do
     Begin
          k:=k+1;
          x:=x div 10;
     End;
       Quantity:=k;
  End;
```

У заголовку функції зазначене її ім'я – `Quantity`. Функції передається тільки один параметр – ціле число, кількість цифр якого треба знайти. Результат – теж ціле число. У розділі змінних описана змінна `k` – лічильник цифр. У тілі функції за допомогою циклу `While` збільшується значення лічильника й щоразу вилучається остання цифра.

Зауважимо, що пам'ять для змінної `k`, яка є локальною, виділяється тільки тоді, коли починає свою роботу функція. Після завершення роботи функції ця частина пам'яті звільняється і значення `k` стає не визначеним. Програма, у якій викликається складена функція, матиме вигляд:

```
Program Example;
Var  n1,n2, k1,k2:integer;
{--------------------------------}
Function Quantity(x:integer):integer;
   Var k: integer;
Begin
      k:=0;
      While x<>0 Do
      Begin
```



```
          K:=k+1;
          x:=x div 10
        End;
     Quantity:=k
  End;
  {--------------------------------}
Begin
     Write('Введіть два числа:');
     Readln(n1,n2);
     k1:=Quantity(n1);{кількість цифр першого
числа}
     k2:=Quantity(n2);{кількість цифр другого
числа}
       if   k1=k2   Then   Writeln('Однакова
кількість цифр')
        Else if k1>k2
  then Writeln('У першому числі цифр більше.')
  else Writeln('У другому числі цифр більше.')
    End.
```

Приклад №2

Знайти найменше спільне кратне трьох чисел.

Розв'язування

Щоб розв'язати цю задачу, використаємо функцію для знаходження найбільшого спільного дільника двох цілих чисел та процедуру для знаходження найменшого спільного кратного двох чисел.

```
Program NSK_3;
Var x,y,z,r1,r2:integer;
{----------------НСД двох чисел-------------}
Function NSD(a,b:integer):integer;
Begin
     While a<>b do
       If a>b then a:=a-b
              else b:=b-a;
  NSD:=a;
End;
{---------------НСК двох чисел -------------}
Procedure NSK_2(a,b:integer; Var Nsk:integer);
```

```
Begin
   Nsk:=(a*b)div NSD(a,b);
End;
{---------- основна програма----------------}
Begin
 Write ('Введіть три цілих числа:');
 Readln(x,y,z);
 NSK_2(x,y,r1);
 NSK_2(r1,z,r2);
 Writeln('NSK_3=',r2);
End.
```

Питання для самоконтролю:

1. *Який загальний вигляд оформлення функцій?*
2. *Як виглядає звернення до функції в головній програмі?*

Заданий фрагмент програми:

```
Var a,b,c,d : Real;
   Function Dum (a,b:Integer; c:Real):Real;
   Begin
      Dum:=a+b*c;
   End;
{----------------------}
Begin
   ...
   a:=Dum(3,6,b);
   d:=Dum(9,12,a);
   ...
End.
```

3. *Який тип має змінна a, яка використовується в операторі Dum:=a+b*c?*
4. *Який тип має змінна a, яка використовується в операторі a:=Dum(3,6,b)?*
5. *Який тип має змінна a, яка використовується в операторі d:=Dum(9,12,a)?*

Дано опис функції:

```
Function S(a,b,c:integer):integer;
Begin
   S:=a*b+b*c+a*c
End;
```

Чому будуть дорівнювати значення змінної z після виконання операторів:

6. *z:=7–S(2,2,3);*
7. *z:=9+S(3,–7,1);*
8. *z:=S(0,14,3)*2;*
9. *Написати текст функції, яка використовується у фрагменті програми:*

```
P:=Per(a,b,c);
WriteLn ('Периметр трикутника=',P).
```

10. *Написати текст функції, яка використовується у фрагменті програми:*

```
m:=Sum(a);
WriteLn('Сума цифр двоцифрового числа',
a,' дорівнює ',m).
```

11. *Написати текст функції, яка використовується у фрагменті програми:*

```
V:=Volume(a,b,c);
WriteLN ('Об''єм акваріума=',V).
```

Задачі

При написанні програм до вказаних нижче завдань обов'язково використовуйте процедури та функції.

1) *Знайти суму цифр цілого числа.*
2) *Знайти суму всіх дільників числа.*
3) *Визначити, чи є число досконалим (використати функцію з попереднього завдання, змінивши при необхідності).*
4) *Відстань між двома точками на площині, заданими координатами, визначається за формулою. Напишіть функцію для знаходження відстані між двома точками. Використовуючи її, знайдіть периметр трикутника, заданого координатами своїх вершин.* $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

2.3. Підготовка до оцінювання теми «Процедури і функції»

Заданий фрагмент програми:

```
Var x,y,z:Real;
   Neo,Tank:Real;
{--------------------}
Function Matrix(a:Integer; b,c:Real):Real;
   Var m,n : byte;
```

```
Begin
   m:= Int(b+c);
   n:= a*a*a-m;
   Matrix:=b/m + c/n
End;
{--------------------}
Begin
...
Neo:= Matrix(15,z,3.14159)+z;
Tank:= Matrix(21,y,0.999)-y;
x:=Neo+Tank;
...
End.
```

1) Вкажіть змінні, які є глобальними.

2) Вкажіть змінні, які є локальними для функції.

3) Вкажіть змінні, які використовуються в якості формальних параметрів функції.

4) Вкажіть дані, які використовуються в якості фактичних параметрів функції.

Дані описи процедури і функції:

```
Procedure First(var a,b:integer;c:integer);
   Begin
      a:=c div a;
      b:=c div b
   End;
{--------------------------------------}
Function Second(m,k:integer):integer;
   Begin
      Second:=m*(k mod 10)+k*(m mod 10)
   End;
```

5) Вкажіть правильні оператори.

а) z:= First (m, k, 32000); б) First (a+1, b–2, 0); в) First (m, k, 99000);
г) Second (7, 8); д) a:= Second (a, b); е) Second (3.14, 8).

6) Вкажіть правильні оператори.

а) d:= First (3, 4, 12); б) First (3, 4, 12); в) First (a, b, c);
г) x:= Second (7, 8); д) Second (m, k); е) Second (a, b).

7) Вкажіть правильні оператори.

а) x:= First (a, b, c); б) First (a–1, b+2, a+b); в) First (a, b, 7);
г) n:= Second (k, m); д) z:= Second (3.14, 8); е) Second (m, k).

Дано описи процедури і функцій:

```
Procedure P (var a,b:integer);
Begin
      a:=a+b;
      b:=a–b
End;
{-------------------------------------}
Function F (m,k:integer):integer;
Begin
      F:=m div 10 + k mod 10
End;
{-------------------------------------}
Function G (a,b:integer):integer;
Begin
      G:=a*a+b*b
End;
```

8) Чому будуть дорівнювати значення змінних a та b після виконання наведеного фрагмента програми?

a:=7; b:=3; P(a,b); b:=F(a,b–a).

9) Чому будуть дорівнювати значення змінних a та b після виконання наведеного фрагмента програми?

a:=98; b:=86–G(a mod 10,a–97); a:=F(a,b).

10) Чому будуть дорівнювати значення змінних a та b після виконання наведеного фрагмента програми?

a:=10; b:=5; b:=G(a,b); P(b,a).

Практичні завдання

11) Знайти найдовшу сторону чотирикутника, заданого координатами своїх вершин у порядку обходу.

12) Задано два натуральні числа M і N. Яке з двох чисел буде більшим, якщо в обох числах переставити крайні цифри?

3. Побудова графічних зображень

3.1. Процедури для оформлення та виведення тексту

Для відображення інформації в середовищі АЛГО виділено робоче поле висотою 2000 точок. Горизонтальний розмір цього поля дорівнює роздільній здатності монітора по горизонталі. Для зручності надалі називатимемо робоче поле **аркушем**. Частину аркуша видно у вікні виведення. За допомогою вертикального та горизонтального повзунків можна переглядати весь аркуш. Виведення текстової інформації та побудова зображень здійснюється на аркуші, а не безпосередньо на екрані монітора.

Для побудови окремих елементів зображення потрібно вказувати їхні координати. Початком координат вважають верхній лівий куток аркуша. Вісь **X** направлена зліва направо, **Y** – зверху вниз. Для всіх графічних операцій першою вказують координату **X**, другою – **Y**.

Якщо координати при побудові зображення виходять за межі аркуша, то інформація за межами аркуша ігнорується. Виняток становлять оператори **ReadLn** та **WriteLn**. При спробі виведення за нижню межу аркуша все зображення посувається вгору на величину висоти символу.

Під час виведення текстової чи графічної інформації необхідно вказати, куди саме виводити: вгорі, посередині чи внизу, а також колір тексту, ліній та заповнення. Вказувати цю інформацію в кожному операторі було б дуже незручно. Тому система запам'ятовує значення координат точки, в якій завершилося виведення інформації та параметри вибраних інструментів. Ці значення називають активними. **У момент запуску програми** на виконання активні координати стають рівні (0,0) (верхній лівий кут екрана), встановлюється чорний олівець одиничної товщини та чорний колір тексту. Колір заповнення замкнутих фігур (колір пензля) вибирається прозорий, тобто замкнуті фігури не зафарбовуються.

В процесі роботи програми активні координати міняються так, щоб наступне виведення починалося там, де закінчилося попереднє.

Активні координати змінюються при виконанні процедур **MoveTo, LineTo, Line, Rectangle, Write, WriteLn, Point, Clear**, а використовуються процедурами **LineTo, Write, WriteLn.**

Активний (поточний) колір тексту встановлюють за допомогою звертання до процедури **КолірТексту**:

```
TextColor (r, g, b:integer);
```

Параметри **r**, **g**, **b** є виразами цілого типу і задають частки червоного, зеленого та синього кольору в результуючому кольорі тексту. Встановленим кольором буде відображатись уся текстова інформація, яка виводиться на екран процедурами **Read** та **Write.** Значення фактичних параметрів при звертанні до процедури **КолірТексту** мають бути в межах від **0** до **255.**

За мовчазною згодою встановлено **чорний колір тексту (0,0,0)**.

Приклад №1

```
Program TextColorDemo;
Begin
  WriteLn('Чорний колір');
  TextColor(127, 127, 127);
  WriteLn('Сірий колір')
end.
```

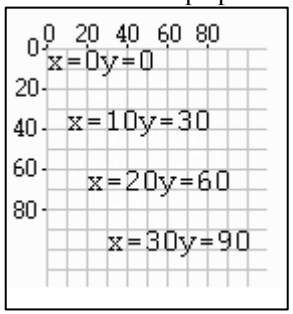
Активна графічна позиція встановлюється в точку з цілими координатами (x,y) процедурою **Перемістити**:

```
MoveTo (x, y:integer);
```

Коли необхідно визначити координати активної графічної позиції, це можна зробити за допомогою процедури **Координати**:

```
Coordinates (var x, y:integer);
```

Фактичними параметрами процедури **Координати** повинні бути змінні цілого типу, яким у результаті виконання процедури надаються значення відповідних координат активної графічної позиції.



Приклад №2

```
Program CoordinatesDemo;
Var i,x,y:integer;
Begin
   For i:=0 to 3 do
    Begin
      MoveTo(10*i,30*i);
      Coordinates(x,y);
```

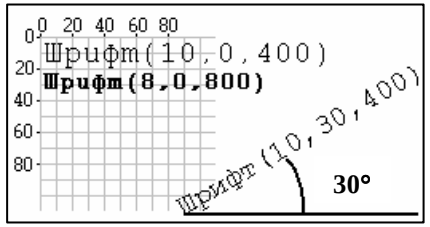
```
    Write('x=',x,'y=',y)
  End;
End.
```

Активний шрифт для виведення тексту встановлюють за допомогою звертання до процедури **Шрифт**:

```
Font (h,a,b:integer);
```

де **h** – висота у пікселах, **a** – кут нахилу в градусах, **b** – насиченість шрифту. Насиченість задають числом в межах від 0 до 1000. Звичайний шрифт має насиченість 400, а жирний – 600. Встановленим шрифтом буде відображатись уся текстова інформація, яка виводиться на екран процедурами **Read** та **Write**.

Фактичними параметрами при звертанні до процедури **Шрифт** мають бути вирази цілого типу.



За мовчазною згодою встановлено шрифт (8,0,400).

Приклад №3

```
Program FontDemo;
Begin
Font(10,0,400);
  WriteLn('Шрифт(10,0,400)');
  Font(8,0,800);
  WriteLn('Шрифт(8,0,800)');
  MoveTo( 80,100 );
  Font(10,30,400);
  WriteLn('Шрифт(10,30,400)');
End.
```

Приклад №4

Розглянемо цікавий приклад поєднання кольорів, шрифтів та переміщення графічного курсора.

Виведемо текст 'I like programming!!!' таким чином:

Розв'язування

Щоб отримати «хвилясте» зображення тексту, яке ви бачите на малюнку, кожен символ ми будемо виводити по графіку функції `sin(i)`. В процедурі MoveTo координата `x` буде збільшуватись виразом `10+i*20` (враховуємо місце для кожної літери і відступ), а для збільшення координати `y` використаємо формулу `y:=round(20*sin(i))`, яка забезпечить «хвилясту» траєкторію. Щоб переглянути в циклі і вивести всі літери тексту, використаємо функцію, яка визначає довжину рядка `length(str)`. При цьому, щоб виділити кожну літеру, вказуємо назву рядка і номер літери `str[i]`. Параметри для процедури `textcolor` задамо з використанням функції `random`, що дасть змогу отримати різнокольорові літери.

```
Program text_demo_1;
Const str='I  like  programming!!!';
Var i,y:integer;
Begin
 Clear;
 For i:=1 to length(str) do
   Begin
    y:=round(20*sin(i));
    MoveTo(10+i*20,100+y);
    Font(10,0,700);
    TextColor(random(200),random(200),random(200));
    Write(str[i])
   end
End.
```

Для витирання аркуша (вікна результатів) використовують процедуру **Стерти** `Clear.` Крім витирання процедура встановлює такі значення активних параметрів:

- чорний олівець товщиною 1;
- чорний колір тексту;
- прозорий колір заповнення;
- активну графічну позицію – (0, 0);
- активний шрифт – (8,0,400).

Іноді необхідно штучно затримати виконання програми на деякий час. Це роблять за допомогою звертання до процедури **Пауза** `Delay (ms:integer).`

Параметр **ms** повинен бути виразом цілого типу і задає число мілісекунд інтервалу чекання. Дана процедура є приблизною, тому період затримки не буде точно рівний заданому числу мілісекунд.

Розглянемо логічну функцію **Подія IsEvent**, яка часто використовується в циклі **Поки** при складанні графічних програм. Подією вважається натискання довільної клавіші на клавіатурі, натискання лівої клавіші мишки у вікні виконання або переміщення мишки з натиснутою лівою клавішею. АЛГО формує повідомлення про подію та запам'ятовує його. Функція **Подія** повертає істинне значення `true`, якщо подія відбулася (наприклад, натиснено клавішу `Enter`), інакше – хибне значення `false`.

Щоб отримати інформацію, яка саме подія відбулася, звертаються до процедури **Повідомлення**

`Event (var ТипПодії, Пар1, Пар2:integer)`.

Ця процедура залежно від типу події присвоює значення параметрам **`Пар1, Пар2.`** Наприклад, якщо тип події дорівнює 2, то це означає, що натиснена ліва клавіша миші у позиції `(X,Y)` і `Пар1` отримує значення координати миші `X`, а `Пар2` значення координати `Y.`

Приклад №5

Вивести текст 'I like programming!!!' так, щоб літери рухались.

Розв'язування

Ефект «руху» виникає тоді, коли ми витираємо зображення і перемальовуємо його знову, змінюючи координати. Додамо в програму з попереднього прикладу цикл **Поки**, процедуру **Пауза** для короткої зупинки (`delay(100)`) та процедуру **Стерти** (`Clear`).

```
While not IsEvent do
   Begin
   …
   Delay(100);
   Clear;
   End;
```

Вся програма матиме вигляд:

```
Program text_demo_2;
Const str='I  like  programming!!!';
Var i:integer;
Begin
 Clear;
 While  not IsEvent  do
 Begin
   For i:=1 to length(str) do
    Begin
      MoveTo(10+i*20,100);
      Font(10,random(10),700);
      TextColor(random(200),random(200),random(200));
      Write(str[i])
    End;
    Delay(100);
    Clear
 End;
End.
```

Питання для самоконтролю:

1. *Яку структуру має вікно виведення інформації?*
2. *За допомогою яких процедур здійснюється виведення тексту?*
3. *Якою процедурою задають колір тексту? Який формат виклику цієї процедури?*
4. *Якою процедурою змінюють активну графічну позицію?*
5. *Яка процедура дає змогу визначити активну графічну позицію?*
6. *За допомогою якої процедури налаштовують шрифт для виведення тексту? Який зміст та типи мають параметри цієї процедури?*

Задачі

1) *Дослідіть роботу програми прикладу 5, замінивши рядок* **`font(10,0,700);`** *на рядок* **`font(10, random(5),700).`** *Встановіть свої кути нахилу тексту.*
2) *Придумайте свій варіант виведення тексту і складіть відповідну програму.*

3.2. Процедури для побудови крапки та лінії

Для виведення крапки, лінії або довільного контуру слід налаштувати інструмент «олівець», який встановлює колір та товщину ліній. Процедура **Олівець** має такий формат:

`Pen (n,r,g,b:integer);`

де **n** – задає товщину лінії в пікселах, а **r**, **g**, **b** – задають частки червоного, зеленого та синього в результуючому кольорі ліній. Встановлені товщина та колір будуть мати усі лінії, які малюються за допомогою процедур **LineTo**, **Line**, **Rectangle**, **Ellipse**, **Point**. Фактичними параметрами при звертанні до процедури **Олівець** мають бути вирази цілого типу, значення яких змінюється в межах від **0** до **255**. За мовчазною згодою встановлено чорний колір ліній (0,0,0) і одинична товщина.

Щоб поставити (намалювати) крапку в точці з координатами (**x, y**), треба звернутись до процедури **Крапка**:

```
Point (x, y :integer);
```

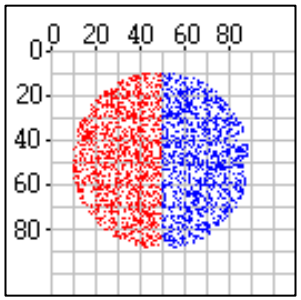
Фактичними параметрами при звертанні до процедури **Крапка** можуть бути змінні, константи чи вирази цілого типу. Колір крапки визначається активним кольором олівця. Після виконання процедури активна графічна позиція встановлюється в точку з координатами **(x,y).**

Приклад №1

Намалювати круг з випадкових крапок. У лівій частині круга крапки мають бути червоного кольору, а в правій – синього.

Розв'язування

```
Program PointDemo;
Var i,x,y:integer;
Begin
  For i:=1 to 5000 do
   Begin
    x:=random(100);
    y:=random(100);
    If sqr(x-50)+
      sqr(y-50)<=1600 then
    Begin
      If x>50 then
       Pen(1,0,0,255)
      else
       Pen(1,255,0,0);
      Point(x,y)
    End
   End
End.
```



Випадкове положення кожної точки забезпечимо випадковим вибором координат X і Y (функція **random(N)**). Якщо вибрані координати потрапляють в круг (що перевіряється рівнянням круга $X^2+Y^2<=R^2$), то виконується додаткова перевірка в ліву (X<50) чи праву (X>=50) його частину. Залежно від цього встановлюється червоний чи синій колір олівця.

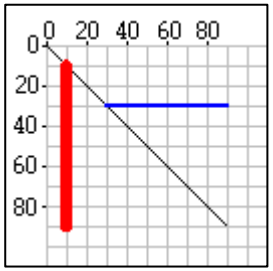
Лінію між двома точками проводять за допомогою звертання до процедури **Лінія**:

```
Line (x1,y1,x2,y2:integer);
```

де **(x1,y1)** та **(x2,y2)** – координати кінців лінії. Фактичними параметрами при звертанні до процедури **Лінія** мають бути вирази цілого типу. Лінія буде проведена активним олівцем. Після виконання процедури активна графічна позиція встановлюється в точку з координатами **(x2,y2)**.

Приклад №2

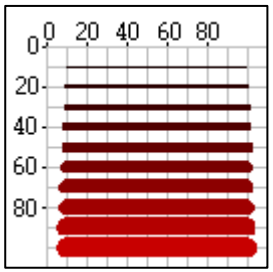
```
Program LineDemo;
Begin
   Line( 0,  0, 90, 90);
   Pen( 2,  0,  0,255);
   Line(30, 30, 90, 30);
   Pen( 6,255,  0,  0);
   Line(10, 10, 10, 90)
end.
```



Розглянемо ще один приклад, який демонструє, як зміна параметрів процедури **Олівець** впливає на товщину та колір лінії.

Приклад №3

```
Program PenDemo;
Var  i:integer;
Begin
   For i:=1 to 10 do
      Begin
      Pen(i, i*20, 0, 0);
      Line(10,i*10,100, i*10);
      End ;
End.
```



Коли наступна лінія починається з кінця попередньої, як буває при побудові ламаних ліній чи графіків, зручно користуватись процедурою **ЛініяДо**:

```
LineTo (x,y:integer);
```

де **(x,y)** – координати кінця лінії. Лінія починається від активної графічної позиції. Фактичними параметрами при звертанні до процедури **ЛініяДо** мають бути вирази цілого типу. Лінія буде проведена вибраним олівцем.

Після виконання процедури **ЛініяДо** активна графічна позиція встановлюється в точку з координатами **(x,y)**.

Приклад №4

```
Program LineToDemo_1;
Var  i:integer;
Begin
 For i:=0 to 100 do
  Begin
   Pen(1,0,2*i+50,2*i+50);
   LineTo(random(100),random(100))
  End
End.
```

За допомогою процедури `LineTo` можна будувати графіки функцій.

Приклад №5

Побудувати графік функції $y = \sqrt{x}$ на відрізку [1,200].

Розв'язування

Залежно від вигляду графіка за допомогою процедури `MoveTo` зміщують точку початку координат. Параметр `i` у процедурі `LineTo` відповідає значенню аргумента, а вираз `50-round(sqrt(i))` значенню функції.

```
Program LineToDemo_2;
Var i:integer;
Begin
  MoveTo(0, 50);
  For i:=1 to 200 do
   LineTo(i, 50-round(sqrt(i)));
End.
```

Питання для самоконтролю:

1. *За допомогою якої процедури встановлюється колір виведення крапки, колір та товщина ліній?*
2. *Яка процедура виводить крапку? Який формат цієї процедури?*
3. *Яка процедура виводить лінію?*
4. *Що можна малювати за допомогою процедури* **`LineTo`***?*

Задачі

1) *Напишіть програму, при виконанні якої за допомогою процедур* **`Line`** *та* **`LineTo`** *малюється кораблик.*
2) *Напишіть програму для побудови графіка функції* $y=x^2$. *Намалюйте вісь X.*

3.3. Процедури для побудови замкнутих контурів

При побудові замкнутих фігур (наприклад, прямокутника, кола) для налаштування вигляду їх заповнення користуються процедурою **Пензель**, яка має вигляд:

```
Brush (k,r,g,b:integer);
```

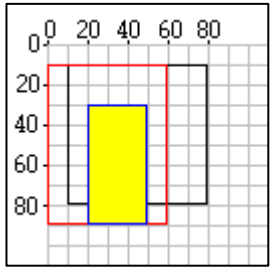
де **r, g, b** – частки червоного, зеленого та синього в результуючому кольорі заповнення, а параметр **k** задає стиль заповнення. Якщо **k=1,** то замкнуті фігури зафарбовуються, а при **k=0** вимальовується лише контур, тобто фігури не зафарбовуються. Встановленим кольором будуть зафарбовуватись фігури процедурами **Rectangle**, **Ellipse** та **Fill**. Фактичними параметрами при звертанні до процедури **Пензель** мають бути змінні, константи чи вирази цілого типу, значення яких знаходиться в межах від 0 до 255.

Прямокутник будують за допомогою процедури

Прямокутник:

```
Rectangle (x1,y1,x2,y2:integer);
```

де **(x1,y1)** та **(x2,y2)** – координати двох діаметрально протилежних вершин прямокутника. Фактичними параметрами при звертанні до процедури **Прямокутник** мають бути вирази цілого типу. Прямокутник буде побудований активним олівцем та зафарбований активним кольором пензля або не зафарбований, якщо встановлений прозорий колір пензля.



Після виконання процедури активна графічна позиція встановлюється в точці з координатами **(x2,y2)**.

Приклад №1

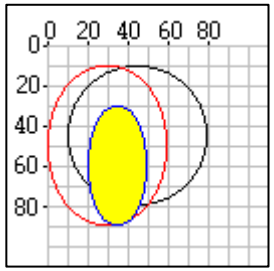
```
Program RectangleDemo;
Begin
   Rectangle(10, 10, 80, 80);
   Pen(1, 255, 0, 0);
```

```
   Rectangle(60, 10, 0, 90);
   Pen(1, 0, 0, 255);
   Brush(1, 255, 255, 0);
   Rectangle(20, 30, 50, 90)
end.
```

Еліпс будують за допомогою процедури **Еліпс**:

`Ellipse (x1,y1,x2,y2:integer);`

де **`x1, y1`** та **`x2, y2`** – координати двох діаметрально протилежних вершин прямокутника, в який вписується еліпс. Якщо прямокутник є квадратом, то еліпс матиме форму кола. Фактичними параметрами при звертанні до процедури **Еліпс** мають бути вирази цілого типу. Еліпс буде побудований активним олівцем та зафарбований активним пензлем або не зафарбований, якщо встановлений прозорий колір пензля. Після виконання процедури активна графічна позиція не змінюється.



Приклад №2

```
Program EllipseDemo;
Begin
   Ellipse(10, 10, 80, 80);
   Pen(1, 255, 0, 0);
   Ellipse(60, 10, 0, 90);
   Pen(1, 0, 0, 255);
   Brush(1, 255, 255, 0);
   Ellipse(20, 30, 50, 90)
End.
```

Щоб зафарбувати замкнуту одноколірну область довільної форми, потрібно звернутись до процедури **Зафарбувати**:

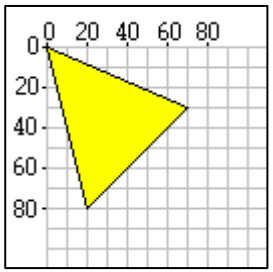
`Fill (x,y:integer);`

Фактичними параметрами при звертанні до процедури **Зафарбувати** мають бути вирази цілого типу. Починаючи з точки з координатами **`(x,y)`**, процедура заповнює екран у всіх напрямках активним кольором пензля до тих пір, поки не зустрінеться межа іншого кольору. Іншими словами, процедура **Зафарбувати** міняє колір замкнутої області, всередині якої лежить точка **`(x,y)`** на активний колір пензля.

Процедура **Зафарбувати** не виконує ніяких дій, якщо заданий прозорий колір заповнення або колір точки **`(x,y)`** співпадає з кольором заповнення.

Приклад №3

```
Program FillDemo;
Begin
   LineTo(70, 30);
   LineTo(20, 80);
   LineTo(0, 0);
   Brush(1,255,255,0);
   Fill(10, 10)
end.
```



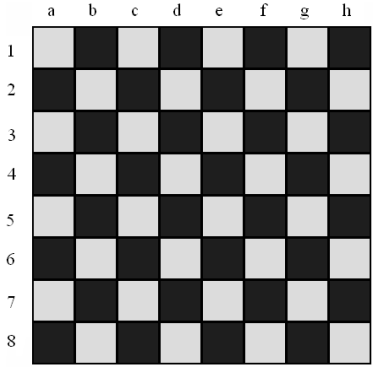
Питання для самоконтролю:

1. *За допомогою яких процедур відбувається заповнення замкнутого контуру? Які особливості оформлення кожної процедури?*
2. *Якою процедурою виводять прямокутник? Який формат виклику цієї процедури?*
3. *Якою процедурою виводять еліпс? Які особливості виклику цієї процедури?*

Задачі

1) *Напишіть програму для малювання 9 кругів (3 рядки по 3 круги в кожному). Зафарбуйте їх так, використовуючи три кольори, щоб в кожному рядку і в кожному стовпчику кольори були різними.*
2) *Напишіть програму, при виконанні якої малюється будинок.*

3.4. Підготовка до оцінювання з теми «Побудова графічних зображень»



Задача. Намалювати шахову дошку.

1-й рівень. Напишіть дві процедури для малювання білого та чорного квадратів.

2-й рівень. Використовуючи дані процедури, намалюйте шахову дошку.

3-й рівень. Зробіть підписи клітинок по вертикалі та по горизонталі.

4-й рівень. Запитайте координати клітинки, в якій треба розмістити фігуру. У вказану клітинку виведіть червоний квадрат.

4. Приклади цікавих програм

4.1. Програма-годинник

Завдання цього параграфа – розробити програму електронного годинника. На прикладі цієї програми ви вивчите роботу процедур опитування часу та методи побудови рухомих зображень.

Для опитування внутрішнього годинника, який є в кожному комп'ютері, використовують процедуру **Time(h,m,s)**, яка надає змінним **h**, **m** та **s** значення години, хвилини та секунди на момент звертання до неї. Досить поставити таке опитування в циклі, щоразу виводити значення часу і годинник готовий.

```
Program Clock1;
Var h,m,s:integer;
Begin
   While not IsEvent do
      Begin
       Time( h, m, s );
       WriteLn( h, m, s );
      End;
End.
```

Якщо запустити таку програму, то відразу видно всі її недоліки. По-перше, кожне нове значення часу виводиться у новому рядку. По-друге, протягом однієї секунди програма декілька разів виводить одне й те саме значення. Так і повинно бути, бо протягом секунди тіло циклу може виконуватись десятки разів.

Щоб позбутися цього, будемо виводити значення часу тільки тоді, коли покази годинника мають змінитися. Для цього введемо нову змінну **sold**, значенням якої буде число секунд, яке вже виведене. Наступне виведення здійснюватимемо лише тоді, коли отриманий час відрізняється від вже виведеного. А перед оператором виведення часу поставимо оператор встановлення позиції виведення, щоб нове значення часу виводилося на тому ж місці, затираючи старе зображення.

Для повноти інформації виведемо ще й дату. Для визначення дати використовують процедуру **Дата**:

```
Date (var Рік,Місяць,День:integer).
```


Процедура **Дата** присвоює фактичним параметрам, які повинні бути **цілими** змінними, значення показників системної дати у момент звертання. Ця процедура використовує три змінні. Але звертання до неї відбувається лише раз, тому ми можемо скористатися тими ж змінними, що й для визначення часу, хоча їхніми значеннями будуть рік, місяць та день календарної дати. Виведемо інформацію в зручнішій формі, і програма може вважатися найпростішим цифровим електронним годинником.

```
Program Clock2;
var h,m,s,sold:integer;
Begin
   Date(h,m,s);
   Write(h,':',m,':',s);
   sold:=0;
   While not isevent do
      Begin
       Time(h,m,s);
       If s<>sold then
         Begin
          Moveto(0,40);
          Write(h:2,' год ',m:2,' хв ',s:2,' сек');
          sold:=s
         End
      End;
End.
```

Практичне завдання

1. Недоліком наведеної програми є те, що виведена дата опівночі не зміниться. Вдоскональте програму, щоб виправити це.

2. Додайте до даної програми графічне зображення календаря, на якому буде виводитись дата, та зображення електронного годинника для виведення часу.

4.2. Інтерпретатор простих виразів

Теорія конструювання мов програмування та перекладу програм з однієї мови на іншу, зокрема машинну, є окремою галуззю науки. У цьому параграфі ви ознайомитесь з елементами процесу розпізнавання і виконання програм на прикладі простого інтерпретатора числових арифметичних виразів.

Розглянемо таку задачу. З клавіатури вводять рядок тексту, який містить цілі числа, розділені знаками арифметичних операцій. Треба обчислити і вивести значення виразу. Для спрощення вважаємо, що рядок не містить пропусків і починається з числа.

Аналіз тексту, який потрібно інтерпретувати (у нашому випадку арифметичного виразу, значення якого треба обчислити), побудуємо таким чином, щоб текст вводився символ за символом і відразу аналізувався. Це дає змогу не тримати в оперативній пам'яті весь текст.

Відомі з арифметики поняття «множник», «ділене», «дільник» об'єднаємо збірним поняттям **множник**. Аналогічно, поняття «зменшуване», «від'ємник», «доданок» об'єднаємо поняттям **доданок**. Отже, вираз складається з доданків а доданки з множників. Будемо розробляти програму так, щоб доданку і множнику відповідали функції, які вводять всі символи даного елементу виразу і обчислюють значення цього елементу. Опишемо змінну `Character`, яка міститиме поточний прочитаний символ.

Розглянемо процес введення числа. Число вводиться цифра за цифрою і введення завершується, коли черговий введений символ не є цифрою. Але ж ми вже ввели його! Вияснити, яким буде наступний символ, не прочитавши його, неможливо. Тому мусимо сформулювати таке правило: процедура чи функція, яка аналізує елемент виразу (доданок або множник), вводить також наступний символ, який вже не належить до даного елементу. Наслідком цього правила є той факт, що перший символ кожного елементу виразу вводиться до виклику функції, яка аналізує цей елемент. На малюнку наведений приклад, який ілюструє це правило.

Товстою лінією підкреслене число **1458**, яке вводиться. Символ **"1"** був введений при попередньому виклику процедури і є значенням змінної **`Character`**. Функція, яка аналізує число, вводить виділені символи, а після її завершення змінна **`Character`** набуває значення **"–"**.

Опишемо необхідні змінні:

```
Program Expression;
Var
```



```
   Character:char;
   Value:real;
```

Напишемо функцію **`Factor`** для введення та обчислення множника. При цьому використаємо той факт, що в таблиці кодування символів цифри упорядковані і мають номери від 48 («0») до 57 («9»). Множники в нашому випадку є цілими числами. Функція **`Factor`** вводить множник цифра за цифрою, поки не зустрінеться символ, який не є цифрою. Наведемо текст цієї функції:

```
Function Factor :integer;
Var  Number:integer;
Begin
   Read(Character);
   Number:=0;
   Repeat
   Number:=Number*10+ord(Character)- Ord('0');
      Read(Character);
   until (Character<'0') or (Character>'9');
   Factor:=Number
end;
```

Напишемо також функцію **`Item`** для обчислення доданка. Функція **`Item`** вводить числа (множники), поки вони сполучені знаками множення та ділення, і виконує відповідні операції:

```
Function Item:real;
Var Number:real;
Begin
  Number:=Factor;
  While (Character='*') or (Character='/') do
    If Character='*' then
        Number:=Number*Factor
    else
        Number:=Number/Factor;
  Item:=Number;
End;
```

Тепер залишилось вводити **доданки**, поки вони сполучені символами додавання або віднімання, і виконувати дії. Зробимо це у виконуваній частині головної програми.

```
Begin
  Value:=Item;
  While (Character='+') or (Character='-') do
    If Character='+' then
        Value:=Value+Item
    else
        Value:=Value-Item;
  Write(Value)
End.
```

Проаналізуємо ще раз роботу всієї програми. У головній програмі вводиться перший символ і викликається функція **`Item`**. Функція **`Item`** звертається до функції **`Factor`**, яка повертає перше число. Якщо далі записано символ множення або ділення, то функція **`Item`** читає наступний символ і знову звертається до обчислення множника. Коли ж зустрінеться будь-який символ крім цифри, «*» і «/», то обчислення доданка припиняється і його значення повертається в головну програму. Головна програма аналогічно формує значення виразу за допомогою операцій додавання та віднімання. Легко переконатись, що виконання програми завершується, коли зустрінеться символ кінця рядка або інший непередбачений символ. Перевірте це!

Практичне завдання

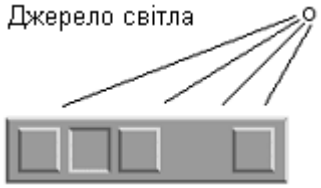
1) *Змініть функцію* **`Factor`** *так, щоб буква «р» сприймалась програмою, як число 3.14.*
2) *Опишіть процес обчислення виразу у вигляді окремої функції* **`Expr`**, *щоб виконувана частина головної програми могла бути записана в такому вигляді:*

```
Begin
  Write(Expr)
End.
```

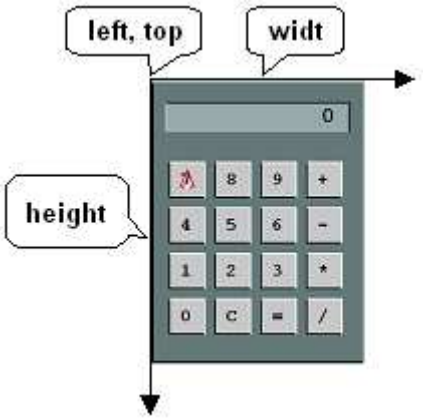
4.3. Проектуємо калькулятор

Працюючи на комп'ютері, ми настільки звикли до різноманітних кнопок і кнопочок на екрані монітора, що використовуємо їх так само легко, як кнопки справжньої клавіатури. При цьому не завжди задумуємось, що вся картина на екрані – це результат роботи програмного забезпечення. Здається, що створити все це надзвичайно складно і така робота вимагає сучасних середовищ візуального програмування. Розглянемо на прикладі розробки програми-калькулятора способи побудови та використання кнопок.

Зображувати та використовувати кнопки на екрані монітора просто і легко. Щоб переконатись в цьому, розгляньте уважно малюнок. Якщо верхню і праву сторони прямокутника, намальованого на однотонному тлі, виділити світлішим кольором, а нижню і ліву – темнішим, то створюється враження, що це – кнопка, яка виступає над поверхнею планки. Поміняємо кольори місцями – кнопка виявиться «натиснутою». При цьому, щоб змінити вигляд кнопки, досить перемальовувати лише межі прямокутника, а його заповнення залишати незмінним. Очевидно, що так легко зображувати можна лише прямокутні кнопки.



Розробимо програму, яка реалізує простий калькулятор для роботи з цілими числами та демонструє використання програмно створених (віртуальних) кнопок.



Щоб почати роботу, програміст має чітко уявляти собі вигляд екрана під час роботи програми. Нехай майбутній калькулятор виглядатиме приблизно так, як показано на малюнку.

Корпус калькулятора, екран, кнопки будемо малювати процедурами, для яких вхідними параметрами будуть координати лівого верхнього кута (`left,top`), ширина (`width`) та висота (`height`) відповідного прямокутного елемента.

Опишемо необхідні змінні:

```
Program Calc;
Const but_size=50;{--Розмір кнопки--}
      but_sym='789+456-123*0C=/';
   {-напиши на кнопках згідно розташування--}
Var i,j,t:integer;
      now_number,save_number,num_add:integer;
      k,x,y:integer;
      operator:char; is_error:boolean;
```

Процедура для малювання корпуса калькулятора:

```
Procedure draw_corpus(left,top,width,height:integer);
Begin
 Pen(1,0,0,0);
 Brush(1,100,120,120);
 Rectangle(left,top,left+width,top+height);
 Pen(3,0,0,0);
 Line(left,top,left,top+height);
 Line(left,top+height,left+width,top+height);
 Pen(3,150,170,170);
 Line(left,top,left+width,top);
 Line(left+width,top,left+width,top+height);
End;
```

Процедура для малювання екрана калькулятора draw_screen містить додатковий параметр для введення числа (number) і перевірки коректності введення даних (err).

```
Procedure draw_screen (left,top,width, height,
number:integer; err:boolean);
Begin
   Pen(1,0,0,0);
   Brush(1,150,170,170);
   Rectangle(left,top,left+width,top+height);
   Pen(2,200,200,200);
   Line(left,top,left,top+height);
   Line(left,top+height,left+width,top+height);
   Pen(2,0,0,0);
   Line(left,top,left+width,top);
   Line(left+width,top,left+width,top+height);
      font(18,0,700);
```



```
    If (err=true) Then
   Begin
    Moveto(left+12,top+2);
    Write(' E R R O R')
   End
   Else
   Begin
    Moveto(left+12,top+2);
    Write(abs(number):12);
   {якщо число від'ємне, то вивести '-'}
    If (number<0) Then
      Begin
       Moveto(left+12,top+2);
       Write('-')
      End
   End
End;
```

Процедура для малювання кнопки:

```
Procedure draw_button (left,top,width,height:integer;
capt:string; is_pushed:boolean);
Var sx,sy:integer;
Begin
   Pen(1,0,0,0);
   Brush(1,200,200,200);
   Rectangle(left,top,left+width,top+height);
{--якщо кнопка натиснута--}
If (is_pushed=true) Then
 Begin
   Pen(2,250,250,250);
   Line(left,top,left,top+height);
   Line(left,top+height,left+width,top+height);
   Pen(2,0,0,0);
   Line(left,top,left+width,top);
   Line(left+width,top,left+width,top+height);
   sx:=7; sy:=13
 End
Else
 Begin
```

```
   Pen(2,0,0,0);
   Line(left,top,left,top+height);
   Line(left,top+height,left+width,top+height);
   Pen(2,250,250,250);
   Line(left,top,left+width,top);
   Line(left+width,top,left+width,top+height);
   sx:=10; sy:=10
 End;
   Font(15,0,700);
   Moveto(left+5+sx,top+sy);
   Write(capt)
End;
```

Наступна функція забезпечує процес введення інформації від користувача до калькулятора. Вона реалізує очікування натискання кнопки миші і повертає номер кнопки, яку натиснув користувач. Очевидно, що така функція повинна повертати ціле значення. Найскладнішим елементом цієї функції є визначення номера **t** кнопки, якій належить точка з координатами **(x,y)**, в якій була натиснута кнопка миші. Для цього ми використаємо стандартну функцію `event(k,x,y)`, яка аналізує натиснення лівої кнопки миші (`k=2`) та дозволяє визначити значення `x` та `y`.

```
Function get_event:integer;
Var i,j,t:integer;
Begin
 get_event:=-1;
 Event(k,x,y);
 if k=2 then
 Begin
   t:=1;
   For i:=1 to 4 do
    For j:=1 to 4 do
      Begin
      If ((x>j*65+10)and(x<j*65+10+but_size)) and
       ((y>i*65+100)and(y<i*65+100+but_size)) then
      Begin
       get_event:=t;
```



```
{---імітуємо зображення натиснення кнопки--}
Draw_button(j*65+10,i*65+100,but_size,but_size,
but_sym[t],true);
       Delay(150);
      End;
      t:=t+1;
      End;
 End;
End;
```

Головна програма матиме такий вигляд:

```
Begin
Clear; Write('Press Esc to Exit.');
now_number:=0; save_number:=0; k:=0; operator:='n';
is_error:=false;
Draw_corpus(50,50,300,400);
While not((k=1)and(x=27)) do
 Begin
   is_error:=false;
   If (now_number>99999999) then
      Begin now_number:=0;
      save_number:=0;
      is_error:=true;
      End;
   Draw_screen(70,80,260,40,now_number,is_error);
   t:=1;
   For i:=1 to 4 do
    For j:=1 to 4 do
      Begin
       Draw_button(j*65+10,i*65+100,
       but_size,but_size,but_sym[t],false);
       t:=t+1;
      End;
   j:=get_event; num_add:=-1;
   Case j of
    1: num_add:=7;
    2: num_add:=8;
    3: num_add:=9;
```

```
    4: Begin save_number:=now_number; now_number:=0;
operator:='+'; End;
    5: num_add:=4;
    6: num_add:=5;
    7: num_add:=6;
    8: Begin save_number:=now_number; now_number:=0;
operator:='-'; End;
    9: num_add:=1;
    10: num_add:=2;
    11: num_add:=3;
    12: Begin save_number:=now_number; now_number:=0;
operator:='*'; End;
    13: num_add:=0;
    14: Begin now_number:=0; save_number:=0;
operator:='n'; End;
    16: Begin save_number:=now_number; now_number:=0;
operator:='/'; End;
    15: Begin
         Case (operator) of
          '+': Begin now_number:=save_number
+now_number; save_number:=0; operator:='n'; End;
         '-': Begin now_number:=save_number-
now_number; save_number:=0; operator:='n'; End;
         '*': Begin now_number:=save_number
*now_number; save_number:=0; operator:='n'; End;
         '/': Begin now_number:=save_number div
now_number; save_number:=0; operator:='n'; End;
        End;
      End;
   End;

   If (num_add<>-1) and (now_number<=99999999)
then now_number:=now_number*10+num_add;
 End;
End.
```

Практичне завдання

1) *Зробіть так, щоб у випадку спроби ділення на нуль калькулятор повідомляв про помилку.*
2) *Додайте кнопки для обчислення потрібних вам функцій.*

4.4. Основи роботи ігрових програм

Ігрові програми – це кара для вчителів інформатики та учнів. Вчителі всіма доступними засобами намагаються заборонити гратися на уроках і дивуються, як без будь-яких пояснень учні засвоюють зовсім не просту систему команд і схему чергової «стрілялки» чи «бродилки». А учням, порівняно з тими «стрілялками», уроки інформатики видаються такими нудними, що не вистачає слів. Давайте поєднаємо приємне з корисним і навчимося програмувати ігри.

Типовий сценарій гри-«стрілялки» можна сформулювати просто: стріляй у все, що ворушиться. Звичайно, у нас не вистачить часу, щоб малювати всіляких монстрів та зброю. Ми реалізуємо лише стрільбу по рухомій мішені і будемо сподіватись, що нестачу художнього оформлення значною мірою компенсує задоволення від того, що програма зроблена самостійно.

Під час роботи ігрової програми на екрані монітора мають бути хоча б два об’єкти: рухома мішень, переміщенням якої керує машина, та приціл, переміщенням якого керує гравець. У момент суміщення їх на екрані гравець має натиснути певну клавішу, щоб здійснити «постріл». Встиг – маєш одне очко, не встиг – починай спочатку.

Реалізуємо спочатку мішень. Нехай це буде червоний кружечок, який рівномірно рухається всередині зеленого прямокутника, як куля без тертя на більярдному столі. Позначимо через **Xm,Ym** координати мішені, через **dX,dY** зміну цих координат за один крок. Щоб створити ілюзію рухомої мішені, треба в циклі перемальовувати мішень щоразу на новому місці. Умовою завершення циклу вважаємо настання якоїсь події. Для того, щоб мішень не «вилетіла» за межі прямокутника, а відбивалася від його країв, будемо міняти знак **dX** на протилежний, якщо мішень наблизиться до бічних стінок, та знак **dY**, якщо мішень наблизиться до верхньої або нижньої стінки. Вважатимемо, що розмір клітки становить 250x250 пікселів і напишемо програму:

```
Програма РухомаМішень;
Змінна
```

```
    Xm,Ym,dX,dY : ціла;
    i,j,k : ціла;
Процедура МішеньЛетить;
Початок
   { витираємо (зеленим кольором) мішень }
   Олівець(1,0,255,0);
   Пензель(1,0,255,0);
   Еліпс(Xm-5,Ym-5,Xm+5,Ym+5);
   { якщо за крок мішень вийде за межі клітки,
      то міняємо напрям руху }
   Якщо (Ym<10) або (Ym>240) то dY:=-dY;
   Якщо (Xm<10) або (Xm>240) то dX:=-dX;
   { знаходимо нове положення мішені }
   Xm:=Xm+dX;
   Ym:=Ym+dY;
   { малюємо (червоним кольором) мішень }
   Олівець(1,255,0,0);
   Пензель(1,255,0,0);
   Еліпс(Xm-5,Ym-5,Xm+5,Ym+5);
   { робимо паузу, щоб побачити мішень }
   Пауза(5);
кінець;
{------головна програма-----}
Початок
   Олівець(1,0,255,0);
   Пензель(1,0,255,0);
   Прямокутник(0,0,250,250);
   Xm:=20;
   Ym:=20;
   dX:=1;
   dY:=2;
   Повторювати
      МішеньЛетить
   докиНе Подія;
кінець.
```

Тепер уже без таких детальних пояснень пишемо програму керування прицілом. Зображуємо приціл у вигляді чорного хрестика, координати якого позначимо через **Xp,Yp**. Для

спрощення не перевіряємо, чи приціл не вийшов за межі прямокутника:

```
Програма КеруємоПрицілом;
Змінна  Xp,Yp : ціла;
      i,j,k : ціла;
Процедура Приціл(X,Y:ціла);
{X,Y – координати, в які потрібно поставити приціл }
Початок
   { витираємо (зеленим кольором) приціл }
   Олівець(1,0,255,0);
   Лінія(Xp,Yp-5,Xp,Yp+5);
   Лінія(Xp-5,Yp,Xp+5,Yp);
   { запам'ятовуємо нові координати }
   Xp:=X; Yp:=Y;
   { малюємо (чорним кольором) приціл }
   Олівець(1,0,0,0);
   Лінія(Xp,Yp-5,Xp,Yp+5);
   Лінія(Xp-5,Yp,Xp+5,Yp);
кінець;
{------головна програма-----}
Початок
   Олівець(1,0,255,0);
   Пензель(1,0,255,0);
   Прямокутник(0,0,250,250);
   Xp:=100; Yp:=100;
   Приціл(Xp,Yp);
   Повторювати
      Повідомлення(k,i,j);
      Якщо k=1 то
       Вибір i із
         37 : Приціл(Xp-5,Yp); { стрілка вліво }
         38 : Приціл(Xp,Yp-5); { стрілка вгору }
         39 : Приціл(Xp+5,Yp); { стрілка вправо}
         40 : Приціл(Xp,Yp+5); { стрілка вниз }
         кінець
   докиНе (k=1) та (i=27) { клавіша Esc }
кінець.
```

Таким чином, обидва об'єкти реалізовані, але окремі. Тепер об'єднаємо їх в одній програмі. Зрозуміло, що в розділі описів треба перелічити всі змінні та помістити обидві процедури.

Виникає ще одна проблема. Приціл повинен весь час перебувати на передньому плані. Якщо мішень пролітає через точку, в якій перебуває приціл, то вона не повинна затирати його. Щоб зробити так, поставимо перед оператором **Пауза** в процедурі **МішеньЛетить** перевірку. У випадку, коли мішень витерла частину прицілу, перемалюємо його. Відповідний оператор матиме такий вигляд:

```
Якщо (abs(Xm-Xp)<10) та (abs(Ym-Yp)<10) то
  Приціл(Xp,Yp);
```

Поставимо першим оператором циклу керування прицілом оператор повторення польоту мішені:

```
Повторювати МішеньЛетить
докиНе Подія;
```

Нехай «постріл» здійснюється клавішею «пропуск», код символу якої 32. Додамо в оператор вибору опрацювання натискання клавіші пробілу. Ми не повинні перевіряти, чи в момент пострілу мішень та приціл мали однакові координати, бо тоді влучити в мішень буде практично неможливо. Досить щоб відстань між ними була меншоюза 10 пікселів.

Наведемо цей фрагмент програми:

```
32 : { натиснений пробіл }
    Якщо(Abs(Xm-Xp)<10) та (Abs(Ym-Yp)<10) то
       Початок
   {Попадання.Великим жовтим кругом показуємо вибух.}
      Пензель(1,255,255,0);
      Олівець(1,255,255,0);
      Еліпс(Xm-30,Ym-30,Xm-(-30),Ym-(-30));
      { робимо паузу, щоб побачити вибух }
    Пауза(50);
    { витираємо жовтий круг }
        Пензель(1,0,255,0);
        Олівець(1,0,255,0);
        Еліпс(Xm-30,Ym-30,Xm-(-30),Ym-(-30));
    { випадкомим чином вибираємо положення та
    швидкість руху нової мішені }
         Xm:=Випадкове(200)-(-20);
```



```
         Ym:=Випадкове(200)-(-20);
         dX:=Випадкове(4);
         dY:=Випадкове(4);
    {відновлюємо приціл, який затерли вибухом}
       Приціл(Xp,Yp);
кінець;
```

Оскільки тема уроку ігрова, то завдань не передбачається. Але можете самостійно підібрати такі значення затримки і швидкостей, щоб було зручно грати. Можете спробувати вести підрахунок числа влучень і промахів. Залежно від кількості пострілів можна збільшувати швидкість польоту мішені. Це в більшості ігрових програм означає рівень. Ви могли помітити, що мішень іноді зупиняється. Виправте цей недолік.

Якщо програма видається занадто складною, можете викинути з неї приціл і керування ним, а «ловити» мішень вказівником миші.

Коли ж все просто і зрозуміло, то можна ускладнити траєкторію польоту мішені. Для цього досить міняти крок переміщення мішені кожного разу на невелику (випадкову чи вибрану іншим способом) величину.

Експериментуйте!

5. Поурочне планування

(32 години + 2 години резервного навчального часу)

№ уро-ку	Тема уроку	Параграф
1. Базові конструкції мови програмування Паскаль		
1	Середовище програмування АЛГО	1.1.
2	Основні елементи мови програмування Паскаль	1.2.
3	Складання найпростіших лінійних програм	1.3.
4	Цілий і логічний типи даних. Оператор розгалуження	1.4.
5	Оператор вибору	1.5.
6	Підготовка до оцінювання з тем «Створення лінійних програм» та «Організація розгалужень»	1.6.
7	Цикл із параметром	1.7.
8	Розв’язування задач з використанням циклу з параметром	1.8.
9	Цикл з передумовою	1.9.
10	Цикл з післяумовою	1.10.
11	Підготовка до оцінювання з теми «Циклічні конструкції»	1.11.
12	Алгоритм Евкліда	1.12.
13	Вкладені цикли	1.13.
14	Дійсний тип даних	1.14.
15	Символьний тип даних	1.15.
16	Рядковий тип даних	1.16.
17	Підготовка до оцінювання з теми «Типи даних»	1.17.
2. Елементи структуризації програми		
18	Підпрограми-процедури	2.1.
19	Підпрограми-функції	2.2.
20	Підготовка до оцінювання з теми «Процедури і функції»	2.3.

3. Побудова графічних зображень		
21	Процедури для оформлення та виведення тексту	3.1.
22	Процедури для виведення крапки та лінії	3.2.
23	Процедури для побудови замкнутих контурів	3.3.
24	Підготовка до оцінювання з теми «Побудова графічних зображень»	3.4.
4. Приклади цікавих програм		
25	Програма-годинник	4.1.
26	Інтерпретатор простих виразів	4.2.
27	Проектуємо калькулятор	4.3.
28	Основи роботи ігрових програм	4.4.
29	Виконання творчих проектів	
30	Виконання творчих проектів	
31	Виконання творчих проектів	
32	Підсумковий урок	

ДЛЯ ЗАМІТОК

ВИДАВНИЦТВО «АСПЕКТ» ПРОПОНУЄ:

Серію посібників для 12-річних середніх навчальних закладів:

- «Інформатика. Початковий курс. 4 клас», Антонова О.П., 2008, – 144 с.
- «Інформатика. Вступ до програмування мовою ЛОГО. 5 клас», Пахомова Г.В., 2008, – 136 с.
- «Програмування мовою ЛОГО. 6 клас», Пахомова Г.В., 2008, – 136 с.
- «Информатика. Основы программирования в ЛОГО. 5 класс», Пахомова А.В., 2008, – 136 с. *РУС*
- «Інформатика. Базовий курс. 7 клас», Шестопалов Є.А., 2008, – 176 с.
- «Базовий курс. 8 клас», Шестопалов Є.А., Сальнікова І.І., 2008, – 216 с.
- «Інформатика. Web-дизайн. 8 клас.», Ковшун М.І., 2007, – 112 с.
- «АЛГО – основи програмування. 8 клас.», Петрів В.Ф., Ріпко Н.А.., 2008, – 104 с.
- «Інформатика. Visual Basic. 9 клас», Бондаренко О.О., 2008, – 200 с.
- «Базовий курс. 9 клас», Шестопалов Є.А., Пилипчук О.П., 2008, – 176 с.
- «Інформатика. Цікаві задачі. 2-9 класи», Антонова О.П., 2008, – 96 с.
- «Мова програмування С++. Спецкурс. 10-12 клас», Лехан С.А., 2007, – 160 с.
- «Turbo Pascal. Спецкурс. 10-12 клас», Бондаренко О.О., 2008, – 272 с.

Серію посібників для 11-річних середніх навчальних закладів:

- «Інформатика. Короткий курс. 10 клас», Шестопалов Є.А., 2008, – 176 с.
- «Короткий курс. 11 клас», Сальнікова І.І., Шестопалов Є.А., 2008, – 208 с.
- «Інформатика. Базовий курс. 10 клас», Шестопалов Є.А., 2007, – 160 с.
- «Інформатика. Практичні та тематичні роботи і проекти. 10-11 класи», Сальнікова І.І., Шестопалов Є.А., 2008, – 160 с.
- «Базовий курс. 11 клас», Шестопалов Є.А., Сальнікова І.І., 2007, – 336 с.
- «Інформатика. Основи алгоритмізації. та програмування. 10 клас», Караванова Т.П., 2008, – 192 с.
- «Інформатика. Збірник вправ та задач з алгоритмізації та програмування. 10-11 класи», Караванова Т.П., 2008, – 152 с.
- «Информатика. Базовый курс. 10 кл.», Шестопалов Е.А., 2007, – 144 с. *РУС*
- «Информатика. Базовый курс, 11 класс», Шестопалов Е.А., Сальникова И.И., 2008, – 320 с. *РУС*

Серію посібників «Для початківця»:

- «Основи комп'ютерної грамотності (Windows'XP, Word'XP, Paint, Internet)», Шестопалов Є.А., 2008, – 176 с.
- «Word'97&2000 для початківця», Шестопалов Є.А., 2008, – 112 с.
- «Excel'2000&XP для початківця», Шестопалов Є.А., 2008, – 112 с.
- «Windows'XP для початківця», Шестопалов Є.А., Моісєєва О.В., 2006, – 160 с.
- «Internet для початківця», Шестопалов Є.А., Ковшун М.І., 2008, – 176 с.
- «Power Point для початківця», Сальнікова І.І., 2008, – 112 с.

Контрольно-діагностичну систему Test-W2

- Друкована інструкція з експлуатації + CD-R: Test-W2 з банком тестів. Алго. ЛогоМиры (Демо). Тренажери. Календарні плани для 7-11 класів тощо.

Для замовлення книг звертайтесь за адресою:

Шестопалов Євген Анатолійович, вул. Тургенєва, буд. 31,
м. Шепетівка, Хмельницька обл., 30400
дом. тел. 8-03840-473-07, моб. тел. 8-066-283-66-18
E-mail: aspekt@sh.km.ua,

Ознайомитися з посібниками і зробити замовлення можна також з мого сайту www.aspekt-edu.kiev.ua

Навчальне видання

Петрів Василь Федорович, Ріпко Наталія Анатоліївна

Інформатика

АЛГО – основи програмування

8 клас

Редактори: *О.О.Бондаренко, О.П. Пилипчук*
Рецензент *П.М.Зінько*
Обкладинка *Є.Ю. Фрейліхман*
Коректор *В.В. Слободян*

Підписано до друку 10.09.2008 р.
Формат 60x84/16. Папір офсетний.
Ум. друк. аркуш 6,5
Зам. Наклад 2000.

Видавець – Шестопалов Є.А.
вул. Тургенєва, буд. 31, м. Шепетівка, Хмельницька обл., 30400
Тел: (03840)-4-73-07, E-mail: aspekt@sh.km.ua
Свідоцтво про внесення до Державного реєстру
суб'єкта видавничої справи ДК № 2170 від 26.04.2005 р.

Шепетівська міжрайонна друкарня.
30400, м. Шепетівка, Старокостянтинівське шосе, 11
Свідоцтво ХЦ № 008 від 9.10.2000 р.
Тел. (03840) 5-15-30