Міністерство аграрної політики та продовольства України
Житомирський національний агроекологічний університет

Бродський Ю. Б., Молодецька К. В.

ІНФОРМАТИКА І ПРОГРАМУВАННЯ

Навчальний посібник

Житомир
2014

Автори: Бродський Ю. Б., Молодецька К. В.

Рецензенти:

І. П. Трофименко – кандидат сільськогосподарських наук, доцент, декан агрономічного факультету Житомирського державного агроекологічного університету;

С. М. Горобець – кандидат педагогічних наук, доцент кафедри прикладної математики Житомирського державного університету імені Івана Франка;

І. І. Сугоняк – кандидат технічних наук, доцент кафедри програмного забезпечення систем Житомирського державного технологічного університету.

Рекомендовано до друку Вченою радою Житомирського національного агроекологічного університету, протокол № 6 від 26 лютого 2014 р.

У навчальному посібнику представлений курс інформатики і програмування, який викладається студентам напряму підготовки 6.080101 "Геодезія, картографія та землеустрій". Розглядається інформатика як наука про інформаційні технології, висвітлені принципи побудови та технології використання комп'ютерних засобів обробки інформації, сучасні інформаційні технології проведення досліджень та загальні підходи до програмування.

Навчальний посібник призначений для студентів вищих навчальних закладів, магістрантів, аспірантів та викладачів.

ЗМІСТ

ПЕРЕДМОВА

Інформатика, як технологічна наука, міцно і назавжди увійшла в життя суспільства. Істотно зростає її роль в процесі впровадження електронних обчислювальних машин в науку і техніку, освіту, управлінську діяльність, виробництво, оскільки сьогодення вимагає від сучасних фахівців уміння своєчасно та якісно опрацьовувати великі потоки інформації в своїй професійній діяльності.

Отже, основним завданням інформатики, як науки, є систематизація прийомів та методів збору, обробки, збереження та передачі інформації засобами обчислювальної техніки.

Суттєвою особливістю сучасної інформатики, на відміну від інших дисциплін, є висока динаміка рширення її предметної області, що вимагає своєчасного врахування науково-технічних досягнень в сфері інформаційних технологій. Крім того, інформатика відноситься до тих фундаментальних дисциплін, що розвивають у студентів такі практичні навички, які відразу знадобляться і в процесі навчання, і в професійній діяльності молодого фахівця.

Виходячи з указаних особливостей інформатики, автори даного навчального посібника "Інформатика і програмування" вважають, що головна мета курсу − надати студентам знання в області сучасних інформаційних технологій, забезпечити фундаментальність освіти майбутніх фахівців, підготувати з них системних аналітиків, здатних приймати важливі, комплексні рішення; допомогти студентам в оволодінні методами та прийомами застосування сучасних комп’ютерних інформаційних технологій, а також навчити самостійно обирати та використовувати сучасний комп’ютерний інструментарій для вирішення фахових завдань.

Вказані особливості визначили структуру і зміст навчального посібника, який складається із 7 розділів. Розділ перший присвячений концептуальним питанням визначення інформатики як науки, розглядаються технічні засоби інформатики – апаратне та

програмне забезпечення персонального комп'ютера. У другому − розглядаються операційні, файлові системи та сучасні новітні технології в інформатиці. Системи обробки текстових і табличних даних (MS Word, Excel), принципи організації баз даних та технології використання відповідного інструментарію (СКБД MS Access) наведені у третьому розділі. Далі (розділ четвертий) викладено фундаментальні підходи до організації комп'ютерних мереж. Основи алгоритмізації та структурного програмування висвітлені у п’ятому розділі на прикладі основних конструкцій алгоритмічної мови програмування Turbo Pascal: простих та складених операторів, стандартних та структурованих типів даних, підпрограм у вигляді процедур та функцій, в тому числі графічних. Математичні методи та інструментарій аналізу даних і моделювання в електронних таблицях наведені у шостому розділі. Заключний сьомий розділ присвячений інструментарію виконання інженерних розрахунків, моделювання та програмування в системі комп'ютерної математики MathCad.

Всі розділи посібника завершуються контрольними питаннями та практичними завданнями, які можна обговорювати та розв’язувати в процесі самостійного опрацювання матеріалу курсу та підготовки до лабораторно-практичних занять.

Автори навчального посібника розраховують, що дане видання буде корисним не тільки студентам указаного напрямку підготовки, а також студентам інших спеціальностей, які вивчають інформатику, магістрантам, аспірантам та викладачам, що використовують сучасні комп’ютерні засоби в своїй професійній діяльності.

Автори висловлюють подяку за сприяння у виданні цього навчального посібника Бродському А. Ю.

СПИСОК УМОВНИХ СКОРОЧЕНЬ

ТР – Turbo Pascal
АЛП – арифметично-логічний пристрій
БД – база даних
ДР – диференціальне рівняння
ЗЗП – зовнішній запамятовуючий пристрій
ІЧ-сигнал – інфрачервоний сигнал
МНК – метод найменших квадратів
НДІ – нормативно-довідкова інформація
НОЗП – надоперативний запам'ятовуючий пристрій
ОЗП – оперативний запам'ятовуючий пристрій
ОС – операційна система
ПЕОМ – персональна електронна обчислювальна машина
ПЗ – програмне забезпечення
ПЗП – постійний запам'ятовуючий пристрій
ПК – персональний комп'ютер
ППП – пакети прикладних програм
СДР – система диференціальних рівнянь
СКБД – система керування базою даних
СП – система програмування

5. ОСНОВНІ КОНСТРУКЦІЇ АЛГОРИТМІЧНОЇ МОВИ PASCAL

У навколишньому світі ми можемо зустріти два радикально протилежні погляди на програмування:
• погляд А: програмування, в основному, досить просте;
• погляд Б: програмування – це дуже складно.
Дейкстра

5.1. Алгоритм та його властивості

Походження терміну "алгоритм" пов’язане із математикою. Слово "алгоритм" з’явилось в результаті перекрученого перекладу із арабської на європейську мову імені узбецького математика, астронома і географа IX ст. Аль-Хорезмі Мухамеда Бен-Муса, який виклав правила арифметичних дій над числами в позиційній десятковій системі числення. Історики вважають, що він був керівником "Будинку мудрості", створеного арабськими халіфами в Багдаді. Його авторству належать роботи з математики і астрономії "Алгебра", "Арифметичний трактат", "Витяг з астрономічних таблиць" тощо. Найважливіша работа з алгебри: "Китай ал-джебр ал-Мухабала" ("Книга про виховання і протиставлення"), де алгебра вперше розглядалась як самостійний розділ математики. Ім'я аль-Хорезмі увійшло в математику як загальна назва будь-якої системи обчислень, що виконуються за строго визначеними правилами – "алгоритмом":

Аль-Хорезмі + аритмос (число) = алгоритм.

Алгоритм – послідовність точно визначених дій для досягнення конкретної мети або розв’язування поставленої задачі. З поняттям алгоритму пов’язані інші поняття: виконувач алгоритму – людина або автомат, який виконує визначений набір дій над даними. Мета виконання алгоритму – отримання результату при визначених початкових даних. Основний принцип побудови алгоритмів полягає в поділі процесу розв’язання задачі на елементарні дії. Розроблений алгоритм повинен відповідати насупним основним властивостям:

1) *зрозумілість* – означає, що виконавець розуміє всі етапи алгоритму;

2) *визначеність (однозначність)* – єдина інтерпретація (тлумачення) правил виконання дій та порядку їх виконання;

3) *дискретність* – можливість представлення алгоритму у вигляді окремих елементарних дій (операцій);

4) *масовість* – складений алгоритм забезпечує розв'язання не однієї окремої задачі, а широкого класу задач даного типу;

5) *скінченність* – завершення роботи алгоритму за кінцеву кількість кроків;

6) *результативність* – означає, що виконання алгоритму повинне закінчуватися отриманням визначених результатів.

Існують такі способи представлення алгоритмів:

а) природна мова (словесне). Приклад: класичний алгоритм Евкліда для знаходження найбільшого спільного дільника двох натуральних чисел.

б) табличний спосіб. Приклад:

A	12
B	8
A-B	12-8
Результат	

в) графічний спосіб (за допомогою блок-схем). Основні умовно-графічні позначення, що використовуються для побудови алгоритму наведені в табл. 5.1.

Базові структури алгоритмів (керуючі структури) – це способи керування процесом обробки даних. Існують три базові структури алгоритмічної конструкції:

1) *лінійна* структура передбачає, що тіло алгоритму представляє собою послідовність команд, що виконуються одна за одною (рис. 5.1, а);

2) *умова (розгалуження)* – це керуюча структура, що передбачає можливість вибору з кількох варіантів, для кожного з яких, залежно від умови, виконується різна послідовність команд, повна форма розгалуження − рис. 5.1, б; неповна форма команди − рис. 5.1, в;

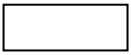
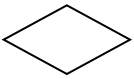
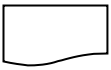
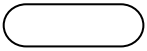
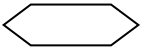
3) *цикл* – це керуюча структура, що дозволяє багаторазово повторювати задану послідовність команд.

Виділяють:

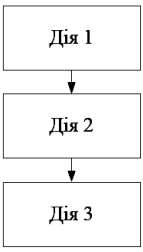
– цикл з передумовою (рис. 5.1, г);

– цикл з післяумовою (рис. 5.1, д).

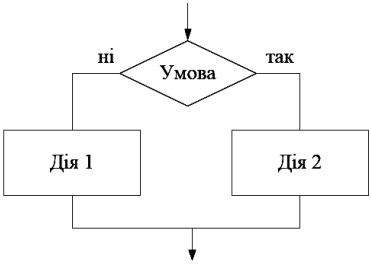
Таблиця 5.1

№	Назва символу	Символ	Відображувана функція
1.	блок обчислень (процес)		обчислювальна дія або послідовність дій
2.	умова		Вибір напряму виконання залежно від умови
3.	блок введення-виведення		введення або виведення даних
			виведення даних на пристрій друку
4.	початок-кінець		початок або кінець програми, вхід або вихід із підпрограми
5.	підпрограма		обчислення за стандартною або користувацькою підпрограмою
6.	блок модифікації		виконання дій, що змінюють пункти алгоритму

При складанні алгоритмів іноді виникає ситуація, коли необхідно виконати повторювану послідовність дій, але не зовсім ідентичну.



а)



б)

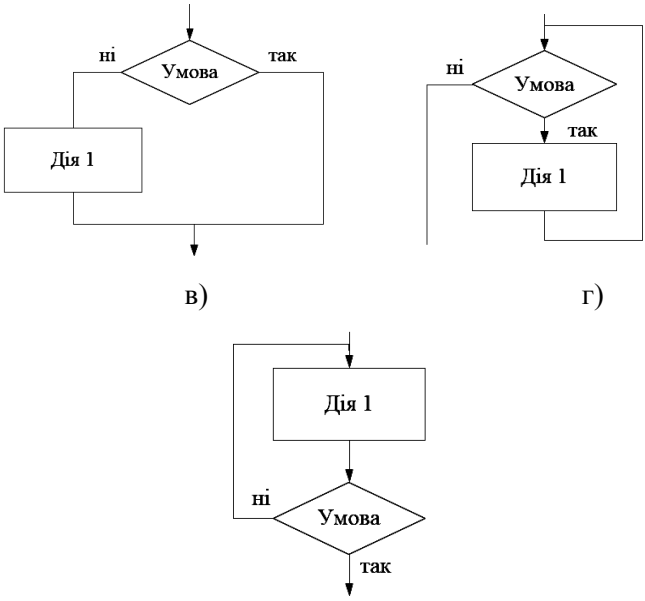


Рис. 5.1.

Приклад. Скласти блок-схему алгоритму обчислення модуля дійсного числа *X*. Розв'язок наведений на рис. 5.2. Щоб не переписувати алгоритми, що суттєво не розрізняються, використовують так звані допоміжні алгоритми, що викликаються і виконуються тільки тоді, коли в них є потреба. Перевага використання допоміжних алгоритмів полягає ще в тому, що склавши їх один раз можна їх потім використовувати при написанні навіть інших алгоритмів, які об'єднуються в бібліотеки.

5.2. Загальні відомості про алгоритмічну мову Pascal

Програма – алгоритм, записаний на мові програмування. *Мова програмування* – сукупність символів (алфавіт), правил утворення (синтаксис) та зміст символьних конструкцій (семантика) для запису алгоритмів. *Транслятор* (від англ. "перекладач") – спеціальна програма, яка переводить текст програми в еквівалентний код (сукупність кодових комбінацій) процесора (на машинну мову).

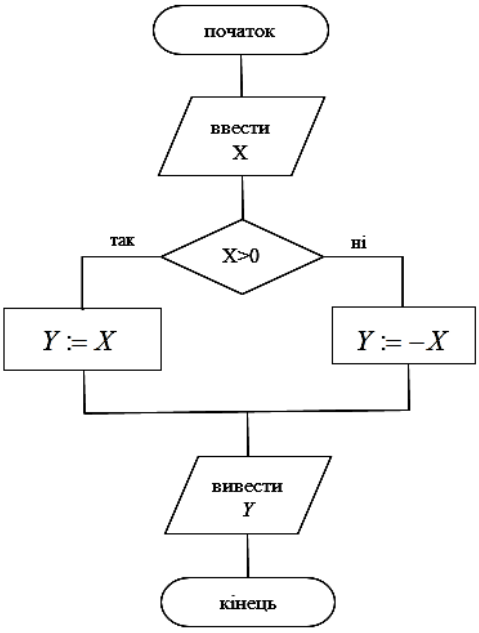


Рис. 5.2.

Існує два способи трансляції: інтерпретація та компіляція (від англ. compile – збирати). *Інтерпретатор* – програма, яка послідовно (відповідно введенню) виконує оператори алгоритмічної мови. *Компілятор* – програма, яка перекладає весь текст задачі (програми) на машину мову. Часто використовують два способи: інтерпретатор – для відлагодження програми, а компілятор – для трансляції відлагодженої програми.

Алгоритмічна мова Pascal (Паскаль) створена професором Цюріхського політехнічного університету Ніклаусом Віртом. Названа на честь відомого винахідника 17 століття Блеза Паскаля. Універсальна алгоритмічна мова Pascal побудована як структурна мова і орієнтована на використання методів структурного програмування. Це досягається за наявності трьох основних компонентів: опис, блочна структура, процедурний апарат.

В описі наводиться інформація для компілятора (імена та типи змінних, мітки, ідентифікатори констант (const), обмеження області значень змінних і тощо).

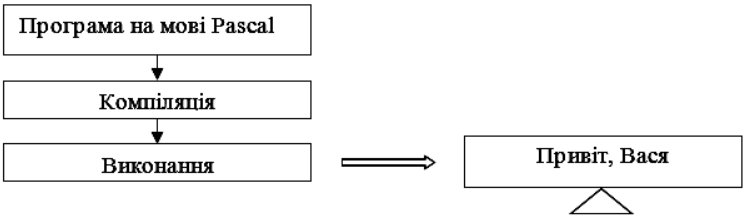


Рис. 5.3.

Блочна структура програми реалізується простими та структурними операторами. Процедурний апарат дозволяє розробляти модульні програми у вигляді ієрархічно організованої сукупності процедур за принципом проектування "зверху донизу", тобто модулі високого рівня визначаються модулями низького рівня.

Лексеми – послідовність допустимих символів мови програмування. Транслятор розглядає програму як послідовність лексем, що поділяються на декілька смислових груп, із яких складаються основні види лексем:

1) Основні символи:

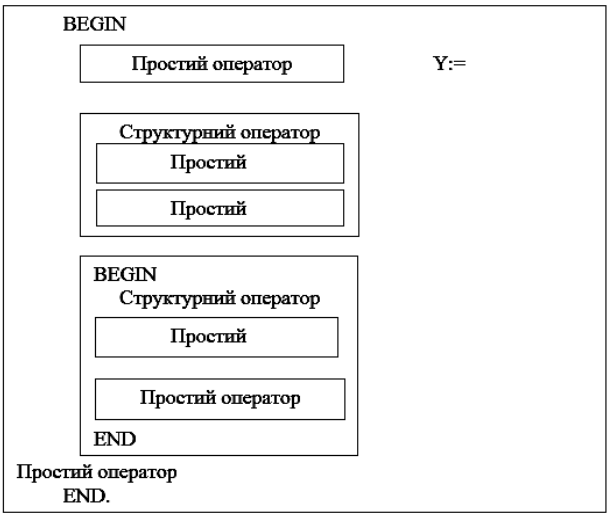


Рис. 5.4. нема посилань

– букви латинського, а також російського алфавіту;

– арабські цифри 0-9.

2) Спеціальні символи:

– арифметичні: +, –, *, /, *div* (цілочислове ділення), *mod* (залишок від цілого числа);

– логічні: ∧ (*and*), ∨ (*or*), ¬ (*not*), <, <=, =, <>, >=;

– присвоювання: :=;

– роздільники: . , : ; - , ?, !, %;

– покажчик: ↑;

– апостроф: ’;

– дужки: ( ) та {коментарій}.

3) Службові слова (зарезервовані) – слова, які компілятор розпізнає без додаткового визначення: *and, array, begin, case, const, div, do, down to, else, end, file, for, function, goto, if, in, label, mod, nil, not, of, or, packed, procedure, program, read, repeat, set, then, to, type, until, var, while, with*.

4) Стандартні ідентифікатори:

– константи: *false, true, maxint*;

– типи: *boolean, integer, char, real, text, string*;

– файли: *input, output*;

– функції: *abs, sqr, sin, cos*, …;

– процедури: *reset* (відкрити файл), *rewrite* (створити і відкрити файл), *read, write, readln, writeln*.

Виділяють наступні основні види лексем:

а) константи: {12; 3,1415926…};

б) імена (ідентифікатори) – позначають деякий об’єкт. (службові слова, стандартні ідентифікатори, імена *const*, змінних, міток, типів, процедур, функцій, модулів, полів у записах);

в) знаки операцій – призначенні для задання дії над операндами (даними) з метою отримання результату;

г) роздільник – виділяють лексеми та інші, більш складні елементи програми: ( ) [ ] , ; : .

д) коментарі – пояснення, що беруться у фігурні дужки {*це коментар*}.

!Особливості!: Пробіл між сусідніми лексемами не обов’язковий, якщо хоча б одна з лексем є роздільником, коментарем, або знаком операції (не іменем). Наприклад, знаки

операцій *div* та *mod* є іменами, а знаки “+” і “–“ не є іменами. Обмеження ідентифікаторів наступне: ідентифікатор складається з латинських букв, цифр, підкреслення, починається тільки буквою, не співпадає із зарезервованими словами, довжина вільна, але значимі перші 63 символи.

Структура Pascal-програми

Загальна структура програми на алгоритмічній мові Pascal подана на рис. 5.5 (а, б).

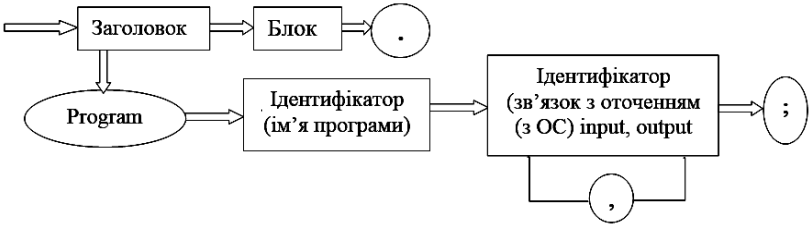


Рис. 5.5, а)

Рис. 5.5, б)

Приклади опису окремих розділів програми наведені нижче.

Опис міток:

Label 1, 4, Lb1, Lb2;

Опис const:

Const P = 3,14159265;
e = 2,71828182;
A = 10;

Типи:

TYPE
mats = array [1…10] of real;
color = (червоний, білий);
index = 0…100;

Змінні:

VAR
A, B, C: *integer;*
D*: real;*

Опис процедур та функцій за структурою подібний до опису програми і починається зі слів: PROCEDURE або FUNCTION.

Оператори:

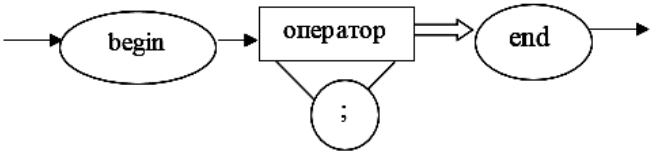


Рис. 5.6.

Приклад: розробити програму для обчислення довжини кола.

```
Program DLINA(input, output);
        const p = 3.14159265;
        var   r, L: real;
begin
        writeln (’Ввести значення радіуса’);
        readln (r);
        L := 2*p*r;
        writeln (L);
end.
```

Розроблена програма починається службовим словом *PROGRAM* та назвою програми: *DLINA*. Після імені у дужках вказаний зв’язок програми зі своїм оточенням – операційною системою: *input* – вказує необхідні дані, які вводитися; *output* – вказує, що програма повинна видати результат з використанням

інструкцій – введення і виведення. Далі опис констант (*const)* та змінних (*var*). Тіло програми: *begin* – службове слово. Оператори розділяються ";". Виконуюча частина програми закінчується службовим словом *end*.

Стандартні функції використовують для перетворення типів даних. Операнд функції (аргумент) записується у круглих дужках: *sin (x)* – повертає значення sin (x); *cos (x)* – повертає значення cos (x); *tan (x)* – повертає значення tg (x); *arctan (x)* – повертає значення arctg (x); *abs (x)* – повертає модуль x; *exp (x)* – повертає значення e^x; *ln (x)* – повертає значення ln x; *sqr (x)* – повертає значення x^2; *sqrt (x)* – повертає значення $\sqrt{x}$; *trunc (x)* – повертає цілу частину x; *frac (x)* – повертає значення x – tranc (x) (дробова частина x); *round (x)* – повертає ціле число найближче до x (округлене); *random (x)* – повертає випадкове число з діапазону $0 \div x$; *odd (x)* – true (істинне) для непарного x, false (хибне) для парного x.

Вирази являють собою формальні правила для виконання дії (обчислень). Взагалі, вирази будуються із: операндів (змінні, масиви, поля записів, вирази функцій); знаків операцій та круглих дужок. Більшість операцій є бінарними, тобто для двох операндів: *a+b*. Однак, є декілька унарних операцій (тобто для 1 операнда), наприклад, *@P*, *-a*. Приклади виразів:

$$(a+b)\cdot c;\ sin\ (x);\ a>2;\ not\,(a>b).$$

Пріоритет операції або оператора – це формальна властивість, що впливає на черговість виконання за відсутності явної вказівки щодо послідовності обчислення. Схема визначення пріоритетів виконання операцій наведена на рис. 5.7.

5.3. Оператори алгоритмічної мови Pascal

Оператори мови Pascal – це синтаксичні конструкції, які призначені для запису алгоритму (перетворення даних, порядок виконання операцій) в стилі структурного програмування. Всі оператори повинні розділятись символом ";". Оператори мови

Pascal умовно поділяють на дві групи: прості оператори та структурні оператори, які містять один або декілька операторів.

Рис. 5.7.

1. Прості оператори:

– оператор присвоювання;

– оператор звертання до процедури (функції);

– оператор безумовного переходу.

Оператор *присвоювання* призначений для обчислення нового значення змінної або функції. Формат задання оператора наведений на рис. 5.8.

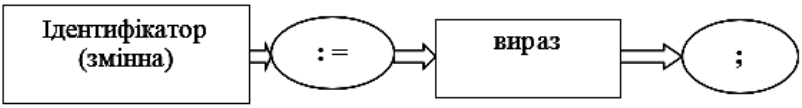


Рис. 5.8.

Оператор виконується так: обчислюється вираз (права частина) і отримане значення надається (присвоюється) ідентифікатору (змінній). При цьому тип виразу (тип даних та

операції над ними) повинен бути сумісним із типом ідентифікатора (змінної), якому присвоюється результат обчислення.

Наприклад, якщо *var* a, b, x: *integer*, то оператор присвоювання $x := a/b$ виконувати не можна. Треба так:

var a, b : *integer*;
x : *real*;

Тоді буде вірним вираз: $x := a/b$.

Приклади:

var name: string; . . . *name : = 'Іванов';*	*var x, y: real;* . . . *x: = 5;* *y: = x + 2;*

Оператори *звертання* до процедури (функції) – призначені для активізації процедури (функції) та передачі їй заданих параметрів. Формат оператора наступний:

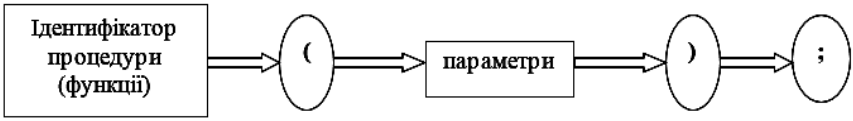


Рис. 5.9.

Розглянемо процедури введення та виведення даних, які можна викликати за допомогою операторів: *read, readln, write, writeln*.

write (вираз), *writeln* (вираз) – оператори виведення;

read (список змінних), *readln* (список змінних) – оператори введення.

Приклади виведення:

1) *x : = 1*;
writeln(1+x);
writeln(x <=1);

на екрані з'явиться:

2
true

2) *x: = 2*;
writeln(1, x, x*x, x*x >2);

на екрані :

1 2 4 true

3) Використання рядкових констант, наприклад: ‘x =’, ‘12’ тощо.

x : = 2;
writeln(‘x = ‘, x,’; y = ‘, x*x);

На екрані:

x = 2; y = 4.

4) Можна вказати розмір поля для виводу значень виразу. Якщо кількість символів менша, то виводяться пробіли, а якщо більша – то виводяться усі символи.

x: =12;
writeln(‘x =’, x : 4,’; y = ‘, x*x : 1);

На екрані:

x =_ _ 12; y = 144 .

Приклади введення:

5) Як правило, для зручності перед оператором введення ставиться оператор виведення на екран запрошення:

writeln (‘Ввести два цілих числа:’);
readln (x1, x2);

На екрані виводиться запрошення: Ввести два цілих числа.

6) Приклад програми привітання.

Program Privit;
var name: *string*;
begin
write*(‘Введіть ваше ім‘я’);*
readln(name);
writeln(‘Привіт,’, name);
end.

Оператор *безумовного переходу* використовується для переходу в задану точку програми без перевірки виконання будь-яких умов (рис. 5.10).

Рис. 5.10.

Мітка описується на початку програми (label 1, 2, lb1) і використовується в програмі як наведено нижче.

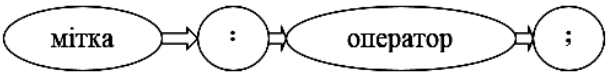


Рис. 5.11.

Однак, використовувати цей оператор в програмі небажано, бо порушується її структурна цілісність і читабельність.

2. Структурні оператори містять один або декілька операторів. Вони поділяються на три види: складений оператор; умовні оператори; оператори циклу.

Складений оператор являє собою послідовність операторів, які розділяються ";" і обмежуються службовими словами *begin* та *end*. Вони відіграють роль дужок обмеження – операторні дужки початку і кінця складеного оператора). Складений оператор сприймається як єдине ціле.

BEGIN
оператор 1;
……….
оператор N
END.

Умовні оператори організовують вибір між альтернативними варіантами обчислень (операцій). Структура оператора:

if <умова (вираз)> ***then*** <оператор1> ***else*** <оператор2>

Семантика оператора:

1) обчислюється умова, тобто вираз після службового слова *if* (результат повинен мати логічний тип, булевий);

2) якщо результат умови – TRUE (істина) то виконується *then* <оператор1>;

3) якщо результат умови FALSE – то виконується *else* <оператор2>.

Оператори 1, 2 можуть бути будь-якого типу: умовні, складені, прості. Умова може бути операцією відношення або логічною. Операції відношення, що використовуються для створення умови:

= , <>, >, <, >=, <=.

Логічні оператори: not – заперечення; and – логічне "і"; or – логічне "або"; xor – "виключне або".

Приклад. Вибір максимального із двох чисел:

if x>y *then* max := x *else* max := y

!Особливість:! При використанні вкладених умовних операторів може виникнути синтаксична неоднозначність. Тому необхідно пам'ятати: службове слово *else* зв'язане з найближчим до нього словом *if*.

Приклад:

```
var a, b, c: integer;
a:=1; b:=2; c:=3; d:=4;
        if a>b then
                if c<d then
                        if c<0 then c:=0
                else a:=b;
```

Результат на екрані: {a = 1}.

```
        if a>b then
                if c<d then
                        if c<0 then c:=0
                          else
                    else
              else a:=b;
```

Результат на екрані: {a = 2}.

Оператор вибору CASE дозволяє вибрати одну з N можливих операцій. Синтаксична діаграма оператора подана на рис. 5.12.

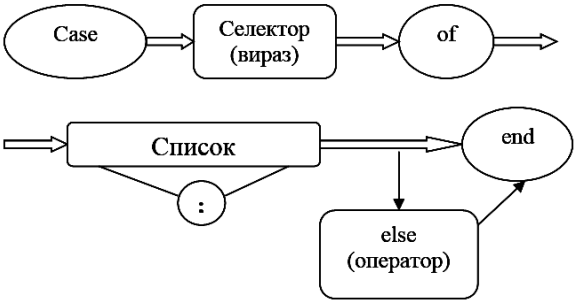


Рис. 5.12.

Список альтернатив задається, як показано на рис. 5.13.

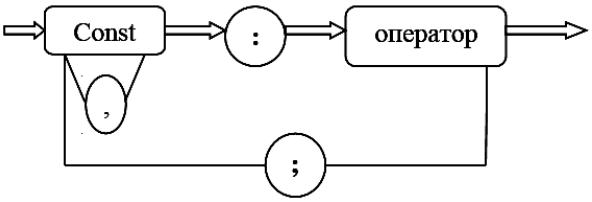


Рис. 5.13.

Приклад:

```
Program month(input, output);
var mon: integer;
begin
        writeln (‘Введіть N місяця’);
        readln (mon);
        if mon < 1 or  mon > 12 then
        writeln (‘Нема такого місяця’)
        else
                case mon of
                        1: writeln (‘січень‘);
                        2: writeln (‘лютий‘);
                        . . .
                        12: writeln (‘грудень‘);
                end
end.
```

!Особливість:! значення селектора не може приймати тип: r*eal*; *string*; частину *else* можна опускати.

3. Оператори *циклу* призначена для організації багаторазового виконання набору інструкцій (команд). Існують три різних оператора: оператор циклу з параметром, передумовою та післяумовою.

Оператор циклу з параметром

Семантика:

1. Обчислення виразу < початкове значення > ;

2. Присвоєння < параметр циклу > : = < початкове значення > ;

3. Перевірка умови < параметр циклу > <= (=>) < кінцеве значення > , якщо не виконується, то *for* закінчує роботу;

4. Виконання < оператора >;

5. Змінна < параметр циклу > на + 1 (*to*) або – 1 (*down to*).

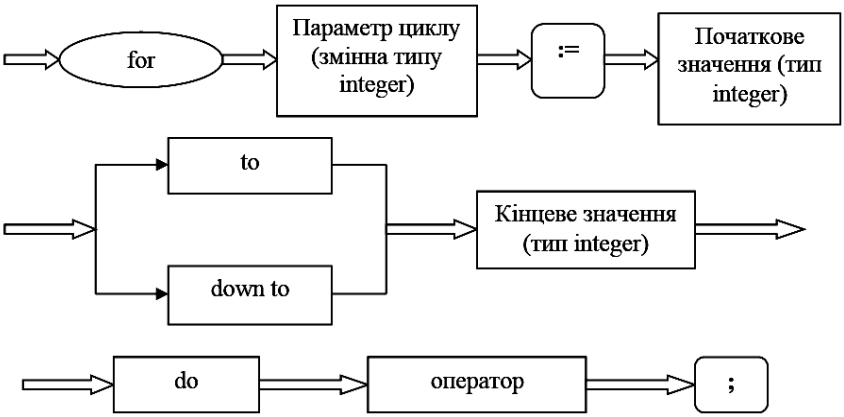


Рис. 5.14.

Приклад: програма введення вільного цілого числа N та обчислення суми цілих чисел від 1 до N.

```
Program SummaInteger;
var i, n, s: integer;
begin
        write ( ‘ n = ‘ );
        readln ( n );                          { ввести n }
        S: = 0;                        { початкове значення суми }
        for i: = 1 to n do s : = s + i;
        writeln (‘Сума дорівнює   ’, S)
end.
. . . . . .
        S:=0
        if n >=1 then
        for:=1 to n do s:= s+i
        else
        for i:= -1 down to n do s:= s+i
```

Оператор циклу з передумовою

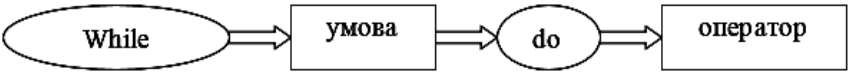


Рис. 5.15.

<умова> – це вираз логічного типу, якщо результат умови TRUE, то виконується оператор, після чого знову проводиться

перевірка умови. Якщо – FALSE, оператор WHILE закінчує свою роботу.

Приклад: обчислити суму перших 25 цілих чисел.

```
Program summa;
var sum, n: integer;
begin
        sum:=0;
        n:=1;
        while n<26 do
        begin
                sum:= sum+ n;
                n:=n+1;
        end;
        writeln (‘Сума перших 25 цілих чисел’, sum)
end.
```

Оператор циклу з постумовою

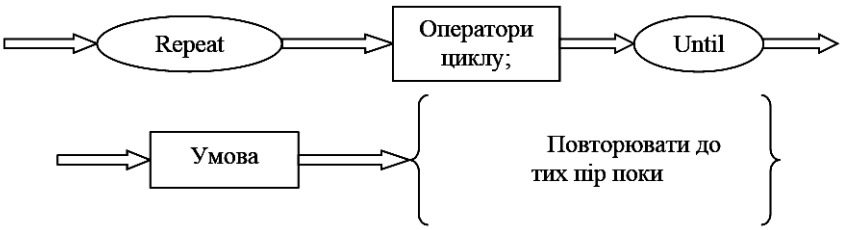


Рис. 5.16.

Семантика: виконується оператор циклу, а потім перевірка умови: якщо *FALSE* => наступне виконання оператора; якщо *TRUE* => оператор циклу не виконується.

Приклад виконання попереднього завдання циклом із післяумовою:

```
repeat
        sum:=sum+n;
        n:=n+1 until n>=26
```

5.4. Типи даних алгоритмічної мови Pascal

Будь-яка константа, змінна, значення функції або виразу характеризується своїм типом. Тип цих об’єктів визначає множину допустимих значень, які може мати об’єкт, а також множину

ІМЕННИЙ ТА ПРЕДМЕТНИЙ ПОКАЖЧИК

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Бродський Ю. Б. Інформатика та системологія: навчальний посібник / Ю. Б. Бродський, К. В. Молодецька; Житомирський національний агроекологічний університет. – Житомир : ЖНАЕУ, 2014. – 246 с.

2. Волков В. Б. Информатика: учебник для вузов / В. Б. Волков, Н. В. Макарова. – СПб. : Питер, 2011. – 576 с.

3. Могилев А. В. Информатика / А. В. Могилев, Н. И. Пак, Е. К. Хённерю. – [3-е издание]. – М. : Академия, 2004. – 848 с.

4. Назаров С. Современные операционные системы / С. Назаров, А. Широков : [Электронный ресурс]. – Режим доступа : http://www.intuit.ru/studies/courses/631/487/info.

5. Котельников Е. Введение во внутренее устройство Windows / Котельников Е. : [Электронный ресурс]. – Режим доступа : http://www.intuit.ru/studies/courses/10471/1078/info.

6. Клементьев И. П. Введение в Облачные вычисления / И. П. Клементьев, В. А. Устинов; Уральский государственный университет им. А. М. Горького. – Екатеринбург : УГУ, 2009. – 233 с.

7. Голицын А. И. Word 2010. Создание и редактирование текстовых документов / П. П. Мирошниченко, А. И. Голицын, Р. Г. Прокди. – М. : Наука и техника. – 192 с.

8. Зудилова Т. В. Работа пользователя в Microsoft Excel 2010 / Т. В. Зудилова, С. В. Одиночкина, И. С. Осетрова и др.; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики. – СПб. : НИУ ИТМО, 2012. – 213 с.

9. Осетрова И. С. Microsoft Excel 2010 для аналитиков / И. С. Осетрова, Н. А. Осипов; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики. – СПб. : НИУ ИТМО, 2013. – 65 с.

10. Сеннов А. С. Access 2010 : [учебный курс] / А. С. Сеннов. – СПб. : Питер, 2010. – 288 с.

11. Олифер В. Г. Компьютерные сети. Принципы, технологии, протоколы : [учебник для вузов] / В. Г. Олифер, Н. А. Олифер. – [4-е

издание]. – СПб. : Питер, 2010. – 944 с.

12. Ларин Рональд У. Инженерные расчёты в Excel [пер. с англ.] / Ларин Рональд У. – М. : Вильямс, 2002. – 544 с.

13. Минько А. А. Статистический анализ в MS Excel [пер. с англ.] / Минько А. А. – М. : Вильямс, 2004. – 528 с.

14. Фаронов В. В. Turbo Pascal. В подлиннике / В. В. Фаронов. - СПб : БХВ-Петербург, 2004. – 1056 с.

15. Культин Н. Turbo Pascal в задачах и примерах / Культин Н. – СПб : БХВ-Петербург, 2006. – 302 с.

16. Немнюгин С. А. Turbo Pascal : практикум / С. А. Немнюгин. – СПб : изд. "Питер", 2001. – 256 с.

17. Потопахин В.В. Turbo Pascal : Решение сложных задач / В.В. Потопахин. – СПб : БХВ-Петербург, 2006. – 194 с.

18. Рапаков Г. Г. Программирование на языке Pascal: [учебное пособие] / Г. Г. Рапаков, С. Ю. Ржеуцкая. – СПб : Питер, 2004. – 480 с.

19. Бродський Ю. Б. Основи використання інструментарію MathCad для математичних розрахунків та моделювання : методичні рекомендації та завдання для самостійної роботи студентів / Ю. Б. Бродський ; Житомирський Національний агроекологічний університет. – Житомир : ЖНАЕУ, 2012. – 91 с.

20. Макаров Е. Г. MathCad : [Учебный курс] / Е. Г. Макаров. – СПб. : Питер, 2009. – 384 с.