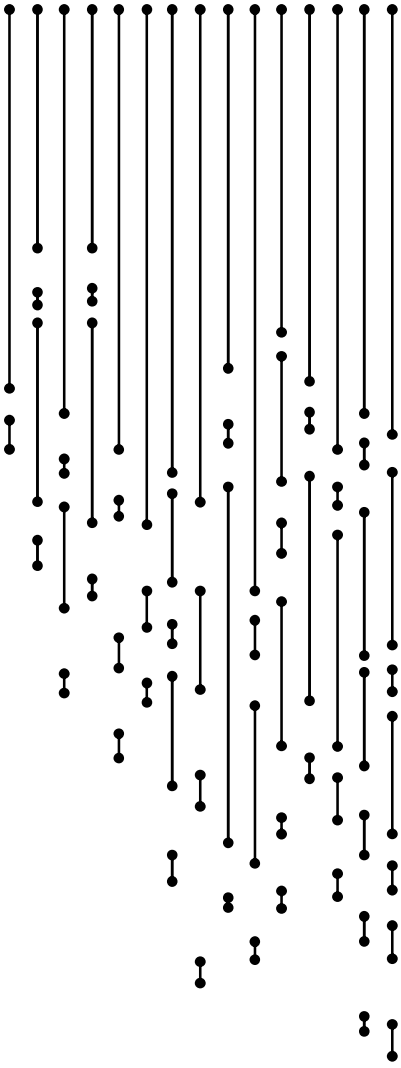
Мельник Л.Є.

Мова програмування

ПАСКАЛЬ

для початківця



Відділ освіти
виконавчого комітету Бібрської міської ради
Бібрський ОЗЗСО І-ІІІ ст. ім. Уляни Кравченко

Мельник Л.Є.

Мова програмування Паскаль для початківця

(*посібник для вчителів інформатики та здобувачів освіти*)

Бібрка 2020

Мова програмування Паскаль для початківця.- Бібрський ОЗЗСО І-ІІІ ст. імені Уляни Кравченко. 2020, - ст.82

Рецензенти:

Мулик Галина Михайлівна*, вчителька інформатики Великоглібовицького ЗЗСО І-ІІІ ст. ім. Ю.Головінського;*

Іванишин Ярослав Михайлович*, вчитель інформатики ЗЗСО І-ІІІ ст. с.Романів*

У посібнику викладений теоретичний матеріал та приклади вправ і задач з мови програмування Паскаль. Наведено опис основних операторів мови, розписано їхню дію та призначення, є опис роботи з одновимірними масивами та рядковими величинами. Подано розв'язки деяких задач із сайту algotester.com, призначеного для підготовки та проведення олімпіад із програмування.

Посібник рекомендовано вчителям та здобувачам освіти, які ще не мають досвіду програмування, бажають опанувати мову програмування Паскаль.

Матеріали схвалено на засіданні вчителів математики, фізики та інформатики Бібрського ОЗЗСО І-ІІІ ст. імені Уляни Кравченко від 18.12.2019р., протокол № 4

Схвалено та затверджено на засіданні методичної ради відділу освіти виконавчого комітету Бібрської міської ради від 24.12.19р., протокол №2.

Редактор *Мельник А.С.*

Відповідальний за випуск *Микитів К.С.*

Комп'ютерний набір *Чабан Х.Й.*

Художник обкладинки *Мартинюк М.Р.*

Комп'ютерна верстка і макетування *Мельник Л.Є.*

Вступ

Мову програмування Паскаль розробив шведський професор Ніклаус Вірт. Перший компілятор мови Паскаль з'явився у 1970 році. Мова названа ім'ям видатного французького філософа і математика Блеза Паскаля, який ще у 17 столітті винайшов і виготовив механічний пристрій для додавання чисел, один із перших калькуляторів.

Мова Паскаль має свої переваги для вивчення основ програмування. Синтаксис Паскаля подібний до звичайної англійської мови, тому його легко зрозуміти навіть для початківця. Мова має строгі вимоги, наприклад, строге дотримання типів даних, перевірка індексу на вихід за межі масиву. Це допомагає запобігти багатьох простих помилок, які часто роблять початківці.

Є різні середовища розроблення програм на Паскалі. Рекомендую використовувати Lazarus.

Lazarus – середовище програмування мовою Паскаль. Перша версія вийшла у 2012 році. Lazarus – це професійне середовище, але воно є простим для встановлення та використання. Сюди входить редактор програм, зневаджувач та багато інших інструментів. Тут можна розробляти як консольні програми, так і програми з графічним інтерфейсом. Це дає великі можливості як для вчителя так і для школяра.

Lazarus є безоплатним і його можна вільно поширювати (на відміну від інших, платних аналогічних програм). Тому це середовище є доступним для всіх, хто хоче навчитися програмувати. Перевагою є також те, що Lazarus підтримує інтерфейс українською мовою. Це робить програмування простішим для вивчення. Крім того, середовище може працювати як під Windows, так і під іншими операційними системами, наприклад, Linux, Mac OS.

Можна розпочати з основних понять мови. Це буде декілька занять теорії без задач. Але сучасні діти не слухають

теорії, тому бажано пояснювати теоретичні моменти лише тоді, коли виникне в цьому потреба.

Ефективно вивчати програмування, використовуючи сайт algotester.com.

Алготестер – це «система автоматичного тестування розв'язків з великим набором цікавих алгоритмічних задач різноманітної складності» (цитата із сайту). Багато задач доступні школярам для розв'язування самостійно. Для початку на цьому сайті треба зареєструватись. Реєстрація і користування сайтом є безоплатним.

Вважаю, що навчити працювати з Алготестером треба на самому початку вивчення програмування, адже там є задачі, які можна використовувати як проблемне питання на заняття.

Знайомство з середовищем

Середовище програмування Lazarus можна завантажити на офіційному сайті. Разом із середовищем буде встановлено компілятор Free Pascal.

Для початку треба зареєструватися на сайті Алготестера. Після реєстрації можна надсилати свої файли на тестування – перевірку на правильність.

Створення програм у Lazarus'і

Розглянемо спосіб створення програм у Lazarus'і.

Щоби створити програму в консольному варіанті, треба запустити Lazarus, а далі виконати такі дії: **Файл → Новий → Програма → Гаразд**. Отримаємо редактор тексту у якому зверху є службовий текст, а знизу **begin end**.

Коли написана програма у Lazarus'і, її треба запустити на виконання клавішею F9 на клавіатурі, появиться чорне консольне вікно виконання програми, де треба ввести вхідні дані та подивитися результати. Вони мають збігатися із тими, що є в прикладі Алготестера.

Зберегти написану програму можна діями **Файл → Зберегти як → відкрити потрібну папку → задати назву файлу англійськими літерами → Зберегти**.

Опісля треба надіслати файл із розширенням .pas на перевірку в Алготестері. Якщо програма дає правильну відповідь на всі тести, які є для цієї задачі на Алготестері і попри те вкладається в час виконання та дозволений обсяг пам'яті, тоді Алготестер пише «Зараховано». Тобто задача розв'язана.

Робота з Алготестером

У Алготестері є задача 0001 «А плюс В». Щоби відкрити умову задачі треба клацнути на назву задачі «А плюс В», а далі перейти на вкладку «Умова»:

А плюс В

Обмеження: 2 сек., 128 МіБ

Дано два цілих числа A та B. Ваше завдання – обчислити їхню суму.

Вхідні дані

Два цілих числа A та B через пробіл.

Вихідні дані

Сума A та B.

Обмеження

$0 \le A \le 100$,

$0 \le B \le 100$.

Приклади

Вхідні дані	Вихідні дані
4 7	11
47 74	121

Примітки

У першому прикладі 4 + 7 = 11.

У другому прикладі 47 + 74 = 121.

Цю сторінку треба докладно обговорити і пояснити кожний рядок окремо, щоб надалі було зрозуміло, що означають такі записи в умові задачі Алготестера.

На початку є назва задачі та сама умова.

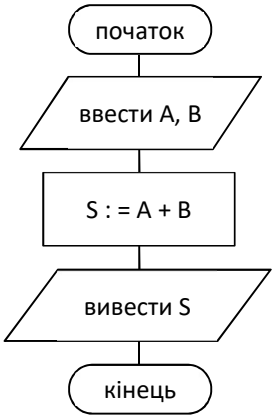
Далі записані «Вхідні дані» – це дані, значення яких будуть вводитися під час виконання програми. У цій задачі вхідні дані – це записані в один рядок через пропуск два цілі числа A та B.

Наступним є розділ «Вихідні дані», де вказано, що треба вивести. Вихідні дані у цій задачі – одне єдине число, сума чисел A + B.

Наступним рядком є «Обмеження», де вказані межі для вхідних даних: $0 \le A, B \le 100$.

Ще нижче є табличка, де є приклади значень вхідних та вихідних даних. Далі можуть бути пояснення, чому для вхідних даних із таблиці отримали саме такий результат.

Графічно розв'язок має такий вигляд:



На Паскалі програма має вигляд

```
Var a, b, S: Byte;
Begin
    ReadLn(a, b);
    S := a + b;
    WriteLn(S);
    ReadLn
End.
```

Розв’язок цієї задачі можна знайти в Алготестері у розділі «Допомога». Там можна прочитати і про роботу із задачами.

Цю задачу треба дати для того, щоб навчитися вводити програму у Lazarus’і, запускати її на виконання, зберігати, надсилати у Алготестер та отримувати «зараховано». Все це може бути як перша практична робота з програмування.

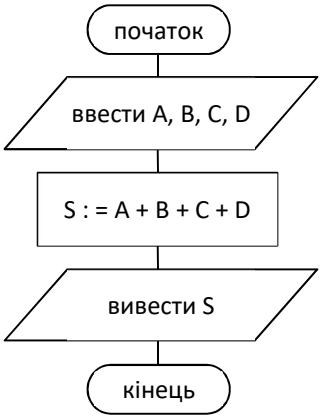
Надалі можна розглянути задачу 0521 «Євро 2012». Аналогічна задача, у якій треба знайти суму чотирьох чисел.

Діти мають самі знайти алгоритм розв’язання і запропонувати зміни до програми написаної для задачі 0001.

Особливістю цієї задачі є те, що тут треба скласти інформаційну модель, бо, на відміну від задачі «А плюс В» тут нема позначень для величин. Насамперед, треба придумати назви вхідних даних. Наприклад, A – число сувенірів для дівчат зі Львова, B – для дівчат із Києва, C, D – з Харкова та Донецька відповідно.

З великого тексту умови задачі визначаємо, що саме треба зробити у програмі. Отримаємо інформаційну модель: дано чотири натуральні числа у межах від 0 до 1000 A, B, C, D, знайти їх суму.

Графічно розв’язок має вигляд



І програма на Паскалі:

```
Var a, b, c, d, S: Integer;
Begin
    ReadLn(a, b, c, d);
    S := a + b + c + d;
    WriteLn(S);
    ReadLn
End.
```

Розв'язавши цих дві задачі, діти уже мають певне уявлення про що йде мова і можуть сприйняти теоретичну основу цих прикладів.

Розділ І. Лінійні програми

Програма мовою програмування Паскаль має такий загальний вигляд:

```
    { розділ описів }
Begin
    { розділ операторів }
End.
```

Слова Begin та End означають «початок» і «кінець». Після останнього End ставимо крапку, що означає кінець програми. Якщо після End. ще щось написано, воно не буде виконуватись.

Розділ описів може містити описи сталих, функцій, підпрограми, міток та змінних.

На початку треба сказати тільки про опис змінних.

Опис змінних

Опис змінних має такий вигляд:

```
Var список змінних: тип змінних;
```

Опис змінних вказує комп'ютеру, які є змінні і скільки потрібно виділити пам'яті для кожної змінної згідно з вказаним типом. Опис змінних – це як купівля квитка на потяг: ми ще не у поїзді, але місця наші вже зарезервовані. Описані змінні можна буде використовувати в розділі операторів.

Розглянемо докладніше кожну частину. Слово Var означає «змінні» (від англ. Variables). Назву змінної записують буквами латинського алфавіту. У назві змінної також можуть бути символ нижнього підкреслення і цифри (цифра не може бути першим

символом). Значення змінних можуть змінюватися під час виконання програми.

Типи цілих чисел

Тип змінної вказує комп'ютеру розмір пам'яті, який треба виділити для змінної. Розмір пам'яті обмежує діапазон допустимих значень для змінної. А також кожний тип має допустимі дії, які можна виконати над його величинами.

Розглянемо основні типи цілих чисел:

Назва типу	Розмір пам'яті	Діапазон значень
Byte (коротке ціле без знаку)	1 байт	0..255
Word (ціле без знаку)	2 байти	0..65535
Integer (ціле зі знаком)	2 байти	-32768..32767
LongInt (довге ціле зі знаком)	4 байти	$\approx -10^9..+10^9$

✱ Розмір типу Integer залежить від компілятора і розрядності процесора і може займати 2, 4 або 8 байтів.

Є ще й інші типи цілих чисел, про них можна прочитати в інтернеті.

Оператори мови

Оператор – це команда, інструкція, вказівка виконати певну дію. Програма складається з послідовності операторів. Комп'ютер виконує тільки те, що йому задають команди програми.

Оператор введення

Використовують для надання змінним конкретних значень. Має вигляд

```
Read(список змінних);
```

– ввести значення змінних або читати з клавіатури значення змінних, або

```
ReadLn(список змінних);
```

– ввести значення змінних і додатково перейти на новий рядок.

Наприклад, у задачі 0001 маємо

```
ReadLn(a, b);
```

Виконується так: на клавіатурі набираємо «4», натискаємо клавішу «пропуск», далі «11», натискаємо Enter. Це означає, що a = 4, b = 11.

Якщо ввести «11», «пропуск», «4», Enter, то отримаємо a = 11, b = 4. Отже, має значення порядок чисел, які вводять, а саме, перше число – це значення першої змінної; друге число – значення другої змінної і т.д.

Оператор присвоєння

Оператор *присвоєння* використовують, щоб надати значення змінним і має вигляд

```
змінна := вираз;
```

«:=» читається «присвоїти» і задає комп'ютеру записати результат обчислення виразу у змінну.

Розглянемо приклади команд і результати їх дії:

a := 5;	Змінна a отримує значення 5.
b = a;	Змінна b отримує таке ж значення як і a, тобто 5.
c := b + a;	Обчислюється 5 + 5 і результат записується в змінну c.
a := a + 1;	Значення змінної a збільшується на 1, тобто, a отримує значення 6.
c := a + b;	У c записується 11.

Вправа 1. Записати значення змінних після виконанні команд:

```
a := 4;
b := a + 3;
c := b;
b := b + 2;
b := b + 2;
b := b + 2;
c := b;
```

Розв'язання. Покажемо кроки виконання у таблиці, де кожна змінна – це окремий стовпчик, у якому записані значення

цієї змінної. Якщо комірка пуста, то значення змінної не змінилося на цьому кроці:

Команда	a	b	c
a := 4;	4		
b := a + 3;		7	
c := b;			7
b := b + 2;		9	
b := b + 2;		11	
b := b + 2;		13	
c := b;			13

Відповідь: a = 4; b = 13; c = 13.

Вправа 2. Записати оператором присвоєння такі команди

а) надати змінній x значення 40;

б) збільшити значення змінної b на 1;

в) змінити значення змінної x на протилежне;

г) змінній c надати значення суми чисел 15 і 3;

д) зменшити значення змінної p на m.

Розв'язок:

а) `x := 40;`

б) `b := b + 1;`

в) `x := -x;`

г) `c := 15 + 3;`

д) `p := p - m;`

Вправа 3. Використати команди присвоєння, щоб поміняти місцями значення змінних a та b.

Ця вправа потребує чіткого розуміння як працюють команди присвоєння. Виконуючи вправу, треба слідкувати за зміною значень у пам'яті. Нехай, a = 3; b = 5. Зазвичай, діти записують a := b; b := a; Коли виконати ці команди, отримаємо

	a	b
a := 3; b := 5;	3	5
a := b;	5	
b := a;		5

Тобто, a = 5 і b = 5. Це не правильно.

Якщо не можуть здогадатися, пропонуємо таке завдання:

Є склянка з чаєм і горнятко з молоком. Як отримати у склянці молоко, а у горнятку чай? Зрозуміло, що треба взяти ще одну посудину. Аналогічно, і в задачі, треба взяти ще одну змінну, наприклад, c. Тоді:

	a	b	c
a := 3; b := 5;	3	5	
c := a			3
a := b	5		
b := c		3	

Вправа 4. Поміняти місцями значення a та b без третьої змінної.

Ця задача була дуже давно на обласній олімпіаді. Це може бути як домашнє завдання. Якщо ніхто не здогадається, треба записати першу команду і після цього дати подумати.

Розв'язок:

```
a := a + b;
b := a - b;
a := a - b;
```

Оператор виведення на екран

Використовують для показу результатів роботи програми. Має вигляд

```
Write(список виведення); або
WriteLn(список виведення);
```

Як і в разі з оператором введення, «Ln» означає перехід на наступний рядок. Список виведення – це

1) сталі і числа чи тексти в апострофах, наприклад:

Оператор	На екрані
`Write(52);`	`52`
`Write('Hello!');`	`Hello!`

2) змінні – виводяться їхні значення, якщо a = 25 то:

Оператор	На екрані
`Write(a);`	`25`

3) числовий вираз – виводиться значення виразу:

Оператор	На екрані
`Write(7 + 3);`	`10`

4) вираз зі змінними – вираз обчислюється і виводиться результат – значення виразу, наприклад, якщо a = 25:

Оператор	На екрані
`Write((a + 5) * 2);`	`60`

Приклади:

Оператор	На екрані
`Write(7, 4);`	`74`
`Write('a + b');`	`a + b`
`Write('3 + 2 = ', 3 + 2);`	`3 + 2 = 5`

Лінійні програми

Лінійна програма містить тільки оператори введення, присвоєння і виведення. Всі команди виконуються тільки один раз у записаному порядку. Їх ще називають програмами слідування:

Наприклад, розв’язок задачі, щоб поміняти значення змінних a та b є лінійною програмою:

```
Var c, a, b: Byte;
Begin
    ReadLn(a, b);
    c := a;
    a := b;
    b := c;
    WriteLn(a, b);
    ReadLn
End.
```

Лінійні програми використовують, наприклад, для виведення текстових повідомлень, обчислення виразів.

Розв’язки задач із Алготестера 0001 і 0512 є прикладами лінійних програм.

Вправа 1. Вивести на екрані у стовпець свої прізвище, ім’я, по батькові англійською мовою.

```
Begin
    WriteLn('Koval');
    WriteLn('Igor');
    WriteLn('Ivanovych');
    ReadLn
End.
```

Вправа 2. Написати калькулятор, який додає, віднімає та множить два цілих числа.

```
Var a, b: Integer;
Begin
    ReadLn(a, b);
    WriteLn(a + b);
    WriteLn(a - b);
    WriteLn(a * b);
    ReadLn
End.
```

Ділення для цілих чисел

Для цілих чисел вводять дві операції, пов’язані із діленням – це «div» – неповна частка та «mod» – залишок від ділення.

Якщо 25 : 3 = 8 (остача 1) то

```
25_div_3 = 8
25_mod_3 = 1
```

`div` і `mod` з обох боків відділяються пропусками.

Приклади:

```
241 div 10 = 24
241 div 100 = 2
241 div 1000 = 0
241 mod 10 = 1
241 mod 100 = 41
241 mod 1000 = 241
```

✱ Для ділення націло ділення косою рискою не використовують. Результат ділення цілих чисел косою рискою не є цілим числом, а дійсним, наприклад 4 : 2 = 2.0

Вправа. Що можна сказати про число a, якщо

1) `a div 100 = 0` (відповідь: a < 100)
2) `a mod 2 = 0` (відповідь: a – парне)
3) `a mod 2 = 1` (відповідь: a – непарне)
4) `a mod 5 = 0` (відповідь: a – кратне 5)
5) `a mod 10 = 1` (відповідь: a закінчується цифрою 1)
6) `a div 100 = 4` (відповідь: a має 4 сотні)
7) `a div 500 = b` (відповідь: a має у собі b разів число 500)

Задача. Фабрика виготовляє олівці і фасує їх в одинакові коробки. Визначити, скільки вийде повних коробок олівців і скільки олівців залишиться.

Задачі з Алготестера. Для вивчення операторів div і mod варто розв'язати задачі «Депутатські гроші» (0021) і «Решта» (0681).

Можна поділити групу на дві частини і організувати як роботу над груповим проектом. Бажано дати дітям час вивчити умову, висловити свої думки, ідеї та пропозиції.

Обидві задачі зводяться до задачі про розмін монет, в цьому разі – розмін грошей: знайти найменшу кількість купюр по 500, 200, 100, 50, 20, 10, 5, 2, 1 гривні, якими можна оплатити суму N гривень.

Цю задачу розв'яжемо «жадібним» алгоритмом. Беремо найбільшу можливу кількість купюр найбільшого номіналу. Для решти, яка залишилася повторюємо те саме. І так доки не наберемо всю суму.

Наприклад, якщо N = 1240, то беремо 2 купюри по 500 грн, так маємо 1000 грн, 240 грн залишається. Далі беремо одну купюру 200 грн, залишається 40 грн. Не беремо жодної купюри номіналом 100 грн і 50 грн, тому що ці купюри більші, ніж 40 грн. І тоді ще беремо дві купюри по 20 грн.

Щоб знайти, скільки треба взяти купюр по 500 грн. Треба виконати операцію

```
a := N div 500;
```

Далі треба від N забрати цих a купюр, залишиться

```
N := N – 500 * a;
```

Аналогічно треба зробити з усіма решта купюрами, наприклад з 200 грн:

```
b := N div 200;
N := N – 200 * b;
```

В кінці треба всі купюри додати

```
a + b + …
```

Під час розв'язування задачі «Депутатські гроші» треба оцінити яких значень набуватимуть a, b… Якщо $N = 10^9$, то a := N div 500 також буде велике число $a < 10^7$, тобто, N, a: Longint, а коли виконати N := N – a * 500; то залишиться менше, ніж 500, тобто буде N < 500. Тому b, ... можуть мати тип Byte.

✱ Для купюр стандартного номіналу жадібний розв'язок працює правильно. Однак, якщо є купюри нестандартного номіналу, то жадібний розв'язок не завжди дає правильний результат. Наприклад, уявімо, що є купюри по 100, 80, 2 грн і потрібно розміняти суму 170 грн. В результаті жадібного алгоритму отримаємо 36 купюр: 1 * 100 + 35 * 2, але цю суму можна набрати всього 7 купюрами, якщо взяти 2 * 80 + 5 * 2. Про інший спосіб розв'язування цієї задачі можна прочитати в інтернеті.

Дійсні типи чисел

Дійсне – це число як десятковий дріб. Ціла і дробова частина відділені крапкою, наприклад, 23.745, -0.29, 5.0.

У мові Паскаль є декілька типів дійсних чисел. Будемо використовувати тип Real. Розмір типу Real залежить від процесора і може займати 4 або 8 байтів. Здебільшого, на сучасних комп'ютерах він займає 8 байтів і має діапазон від $5 \cdot 10^{-324}$ і до $1.7 \cdot 10^{+308}$ і такий же діапазон для від'ємних чисел. Хоч діапазон великий, однак цей тип має тільки 16 правильних розрядів, а всі нижчі розряди замінені на нулі.

Дійсні числа можна подати у вигляді з фіксованою крапкою, де крапка стоїть між цілою і дробовою частинами, а також у

вигляді з рухомою крапкою ±mE±p. Тут m – мантиса числа, p – порядок числа.

Наприклад:

```
3.2E+29
-9.415E-25
```

Щоб записати такі числа у вигляді з фіксованою крапкою, треба перенести її на p знаків вправо, якщо $p > 0$ і вліво, якщо $p < 0$.

Розглянемо приклади:

```
2.4E+2 = 2.400E+2 = 240.0
-1.049E+4 = -1.04900E+4 = -10490.0
3.51E0 = 3.51
2.4E-2 = 002.4E-2 = 0.024
-1.049E-4 = -00001.049E-4 = -0.0001049
```

Якщо a – число типу Real, то Write(a) виводить число у вигляді з рухомою крапкою. Щоб побачити його з фіксованою крапкою, треба записати після a формат виведення :m:n, де m означає скільки знаковість займає все число, а n – скільки буде цифр після крапки, остання цифра буде заокруглена.

Наприклад, якщо

```
a = 72.12345678
```

Тоді оримаємо

Оператор	На екрані
`WriteLn(a:8:2);`	`␣␣␣72.12`
`WriteLn(a:8:4);`	`␣72.1235`
`WriteLn(a:8:0);`	`␣␣␣␣␣␣72`
`WriteLn(a:2:0);`	`72`

Числові функції

В Паскалі є різні функції для різних типів даних. Є функції для цілих і для дійсних чисел. Потрібно звертати увагу як на тип даних аргументу, так і на тип результату.

Ось деякі основні математичні функції, де x може бути ціле або дійсне число:

Функція	Математичний вираз	Тип даних результату
ABS(x)	$\|x\|$	Такий самий як x
SQR(x)	x^2	Такий самий як x
SQRT(x)	$\sqrt{x}$	Дійсне
SIN(x)	sin x	Дійсне
COS(x)	cos x	Дійсне
EXP(x)	e^x	Дійсне
LN(x)	ln x	Дійсне

Арифметичні вирази

Арифметичні вирази можуть складатися з чисел, змінних, арифметичних дій над ними, круглих дужок і функцій. Під час написання виразів треба пам'ятати такі правила:

1) Всі символи пишуть в один рядок, нема ні підрядкових (x_1), ні надрядкових (x^2), наприклад:

x_1 = `x1`;
x^2 = `x * x`, або
x^2 = `SQR(x)`;

2) Дроби пишуть в один рядок і використовують дужки:

$\frac{a}{b}$ = `a / b` ;
$\frac{a+1}{b+1}$ = `(a + 1) / (b + 1)`
$\frac{2a}{5bc}$ = `2 * a / 5 / b / c`, або
$\frac{2a}{5b}$ = `(2 * a) / (5 * b * c)`;

3) Знак множення не опускають:

$2ab$ = `2 * a * b`;

4) Аргументи функцій пишуть в дужках:

$\sin 2$ = `sin(2)`;
$\sqrt{4a}$ = `SQRT(4 * a)`;

5) Кожна відкривальна дужка у виразі має мати закривальну дужку:

$\sqrt{|x|}$ = `SQRT(ABS(x))`;

$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ = `SQRT(SQR(x1 - x2) + SQR(y1 - y2))`.

Практична робота. Обчислення значень виразів

Вирази для цієї практичної роботи мають мати зміст для всіх можливих значеннях змінних. Наприклад, a + b можна обчислити завжди, а вираз a / b не має змісту, якщо b = 0.

Розглянемо зразок, за яким буде виконуватися практична.

Завдання. Знайти значення виразу

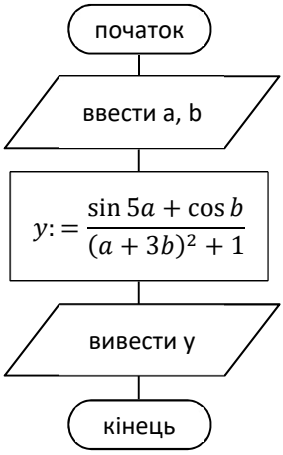
$$\frac{\sin 5a + \cos b}{(a + 3b)^2 + 1}$$

Перевірити правильність програми за допомогою тестових даних.

Аналіз завдання. Оскільки функції $\sin x$ і $\cos x$ визначені для всіх значень x, а знаменник ≥ 1, то цей вираз має зміст для всіх a, b.

Вхідні дані: a, b. Вихідні дані: y – значення виразу.

Блок схема алгоритму:



Підберемо *тестові дані* – це такі значення вхідних даних a, b, для яких ми знаємо, який має вийти результат y. Наприклад, підберемо такі значення, щоб можна було усно обчислити sin і cos. Візьмемо, a = 0 і b = 0, підставимо у вираз:

$$y := \frac{\sin 5a + \cos b}{(a+3b)^2 + 1} = \frac{0+1}{0+1} = 1$$

Тест:

a	b	y
0	0	1

Якщо після виконання програми отримаємо 1, якщо вхідні дані 0 і 0, то робимо висновок, що програма працює правильно.

Програма:

```
Var a, b, y: Real;
Begin
    ReadLn(a, b);
    y := (sin(5 * a) + cos(b)) / (SQR(a + 3 * b) + 1);
    WriteLn(y: 8: 4);
    ReadLn
End.
```

Приклади для практичної:

1) $4mc - 5tc + mt$
2) $4mx^2 - \frac{m-5x}{1+m^2}$
3) $a + 3m - 5c$
4) $52x + 4m + 3xmt$
5) $(8cn - 3)b + 2cnb$
6) $15a - 3(b + 4c)$
7) $p(3 - x + 4t)$
8) $5(a - 3b + 4abp)$
9) $10(c + 5m + 2cx)$

Задачі з Алготестера

На тему лінійних програм в Алготестері можна запропонувати розв’язати такі задачі:

«Торт для Петрика» (0191),

«47-й день року» (1051),

«Агафія та змійка» (1501),

«Автомобіль Санта Клауса» (1535).

Висновок

Лінійні алгоритми можна записати програмою з використанням трьох операторів: введення даних, присвоєння та виведення даних.

У лінійній програмі кожний записаний оператор обов’язково виконується і всього один раз. Задачі на лінійні програми є найпростішими. Однак багато із них вимагають вміння аналізувати, висувати ідеї та логічно мислити.

Розділ ІІ. Програмування алгоритмів з розгалуженнями

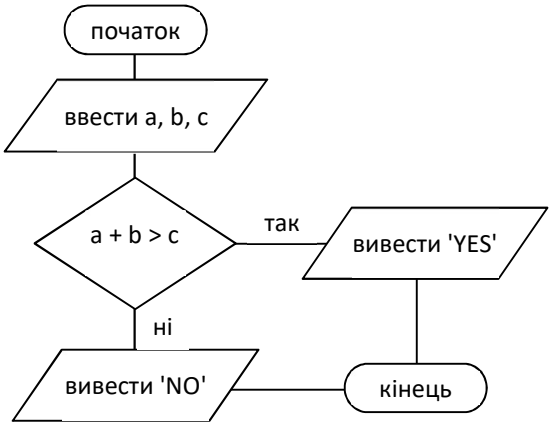
Якщо у задачі є декілька шляхів вирішення, які залежать від певних умов, то вони описуються алгоритмами з *розгалуженнями*. Розгалуження починається з *умови*. Якщо умова виконується («істина»), тоді виконується одна дія, якщо ні («хиба») – тоді інша.

Прикладами застосування розгалуження є розв'язування квадратного рівняння (згадайте, як шукають розв'язок залежно від значення дискримінанта).

Розглянемо задачу «Апельсини» (0481) із Алготестера.

Інформаційна модель цієї задачі така: дано числа a, b, c (a, b, c $\leq 10^9$); вивести 'YES', якщо a + b > c і 'NO', якщо a + b $\leq$ c.

Блок-схема розв'язку:



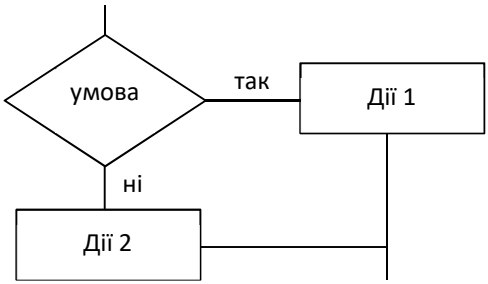
У ромбі записується умова. Умова для певних значень змінних є істинна (так), а для інших значень – хибна (ні).

Розглянемо деякі випадки вхідних даних:

Значення змінних			Умова	Значення умови
a	b	c	a + b > c	-
4	1	3	4 + 1 > 3	істина
4	1	5	4 + 1 > 5	хиба
7	2	10	7 + 2 > 10	хиба

Якщо умова істинна, то комп'ютер виконує тільки дії, записані на блок-схемі у напрямку «так», а ті, що у напрямку «ні» не виконує. Якщо ж умова хибна, то виконує дії у напрямку «ні», а у напрямку «так» не виконує.

Тут використано такий блок розгалуження:



Проста умова – це відношення, записане операторами порівняння >, <, =, <> (не дорівнює), >= (більше або рівне), <= (менше або рівне) між двома виразами, сталими (числами), змінними.

Приклади простих умов:

```
a > 3;
c <= b;
a * 3 <> b – 5 * c.
```

Складені умови

Якщо треба одночасно виконати декілька умов (наприклад, щоб a було найбільшим серед чисел a, b, c, треба зробити два порівняння), використовують *складені умови*.

Умова є складеною, якщо у ній дві або більше умов об'єднанні логічними операціями AND, OR, NOT.

Нехай A, B – деякі умови. Умова (A) AND (B) є істинною тоді і тільки тоді, коли одночасно умови A, B є істинними:

A	B	(A) AND (B)
істина	істина	істина
істина	хиба	хиба
хиба	істина	хиба
хиба	хиба	хиба

AND (анг. «і») позначає *логічне множення*.

Розглянемо приклади:

1) нерівності типу $a \le x \le b$ записують на Паскалі через логічне множення:

```
(a <= x) AND (x <= b)
```

2) $7 < x < 25$ на Паскалі:

```
(7 < x) AND (x < 25)
```

3) Умова «число a найбільше серед a, b, c» має вигляд

```
(a > b) AND (a > c)
```

4) Умова $x \in (7; 19)$ має такий вигляд

```
(7 < x) AND (x < 19)
```

Вправа. Записати декілька значень змінних, для яких умова є істинна; хибна.

1) `(b < a) AND (a <= c)`

a	b	c	Умова	Значення умови
2	1	5	(1 < 2) AND (2 <= 5)	істина
4	0	10	(0 < 4) AND (4 <= 10)	істина
3	5	5	(5 < 3) AND (3 <= 5)	хиба
20	21	10	(21 < 20) AND (20 <= 10)	хиба

2) `(3 * x < 0) AND (x > -3)`

Умова (A) OR (B) є істинною, коли одна із умов A, B або обидві умови A, B є істинні.

Умова (A) OR (B) є хибною тоді і лише тоді, коли обидві умови A, B є хибними:

A	B	(A) OR (B)
істина	істина	істина
істина	хиба	істина
хиба	істина	істина
хиба	хиба	хиба

Англійське слово OR тут використовується у значенні «або». Цю дію називають *логічне додавання*.

Приклади.

1) $x > 30$ або $x < 10$ запишемо

```
(x > 30) OR (x < 10)
```

2) $x \in (-\infty; 3) \cup [15; +\infty)$ запишемо

```
(x < 3) OR (x >= 15)
```

Вправа. Записати значення змінних, для яких умова є істинна; хибна:

1) `(x > 30) OR (x < 10)`

якщо x = 50, тоді (50 > 30) OR (50 < 10) = істина OR хиба = істина;
якщо x = 20, тоді (20 > 30) OR (20 < 10) = хиба OR хиба = хиба;

2) `(a > b) OR (a > c)`

a	b	c	a > b	a > c	(a > b) OR (a > c)
2	3	0	2 > 3 (хиба)	2 > 0 (істина)	хиба OR істина (істина)
3	3	2	3 > 3 (хиба)	3 > 2 (істина)	хиба OR істина (істина)
4	5	7	4 > 5 (хиба)	4 > 7 (хиба)	хиба OR хиба (хиба)

3) `(x < 3) OR (x >= 15)`

Умова NOT (A) істина, коли умова A хиба і навпаки, NOT (A) хиба, коли A істина:

A	NOT (A)
істина	хиба
хиба	істина

NOT (англійською «не») – це *логічне заперечення*.

Наприклад,

$x \neq 0$ запишемо `NOT (x = 0)`;

$a \leq 0$ запишемо `NOT (a > 0)`.

Вправа. Записати значення *x*, для якого умова є істинна; хибна.

1) `NOT (x = 0)`

x	x = 0	NOT (x = 0)
5	5 = 0 (хиба)	NOT (5 = 0) (істина)
0	0 = 0 (істина)	NOT (0 = 0) (хиба)

2) `NOT (a > 0)`

Оператор розгалуження

Повернемося до задачі з Алготестера «Апельсини». Як записати структуру розгалуження на Паскалі? Для цього є оператор розгалуження:

```
If умова Then дія 1 Else дія 2;
```

Читаємо: «якщо умова істинна, виконати дію 1, в іншому разі дію 2».

Виконується розгалуження так:

1) перевіряється умова;

2) виконується дія 1, якщо умова істинна, а дія 2 не виконується;

3) виконується дія 2, якщо умова хибна, а дія 1 не виконується.

Отже, під час виконання розгалуження є такі оператори, які в програмі записані, але не виконуються, залежно від виконання умови.

Запишемо розв'язок задачі «Апельсини»:

```
Var a, b, c: LongInt;
Begin
    ReadLn(a, b, c);
    If a + b > c Then WriteLn('YES') Else WriteLn('NO');
    ReadLn
End.
```

Завдання. Підібрати тестові дані до цієї задачі.

Для розгалужених алгоритмів тестові дані мають передбачати всі варіанти «хиба» та «істина» для всіх умов. У цій програмі умова є одна (a + b > c), отже треба два набори тестових даних.

Маємо:

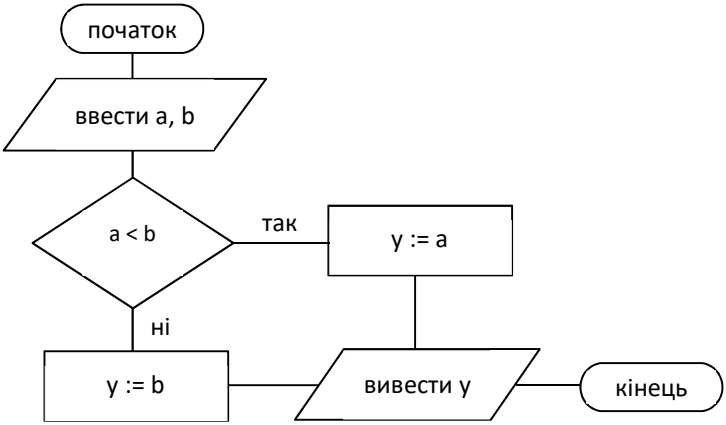
a	b	c	Умова a + b > c	На екрані
5	3	6	5 + 3 > 6 (істина)	YES
4	2	9	4 + 2 > 9 (хиба)	NO

Задачі на розгалуження

Задача 1. Знайти менше з двох натуральних чисел.

Розв'язання. Математичну модель можна записати y = min(a, b), де a, b – дані числа, y – результат.

Блок-схема задачі:



Програма:

```
Var a, b, y: Word;
Begin
    ReadLn(a, b);
    If a < b Then y := a Else y := b;
    WriteLn(y);
    ReadLn
End.
```

Задача 2. Змінити дану програму так, щоб отримати найбільше з двох чисел.

Відповідь. Достатньо в умові записати a > b.

Задача 3. Дано три додатні числа. Дати відповідь чи можуть вони бути сторонами трикутника?

Розв'язання. Відомо, що сума довжин двох сторін має бути більшою за третю сторону. Нехай a, b, c – дані числа (вхідні дані). Тоді умова існування трикутника буде $a + b > c$ і $a + c > b$ і $b + c > a$.

Програма має вигляд:

```
Var a, b, c: Real;
Begin
    ReadLn(a, b, c);
    If (a + b > c) And (a + c > b) And (b + c > a)
    Then WriteLn('YES')
    Else WriteLn('NO');
    ReadLn
End.
```

Задача 4. Дано натуральне число. Дати відповідь, чи воно є парне?

Розв'язання. Парні числа кратні двом, тобто діляться на 2 без остачі. Нехай число позначимо n, тоді ознака парності $n \bmod 2 = 0$.

Програма:

```
Var n: Word;
Begin
    ReadLn(n);
    If n mod 2 = 0
        Then WriteLn('YES')
        Else WriteLn('NO');
    ReadLn
End.
```

Задача 5. Знайти значення виразу

$$\frac{2+x}{x^2-9}+\sqrt{x+3}$$

Розв'язання. Щоб цей вираз мав зміст, треба одночасне виконання двох умов $x^2-9\neq 0$ і $x+3\geq 0$. Програма має вигляд:

```
Var x: real;
Begin
    ReadLn (x);
    If (x * x - 9 <> 0) And (x + 3 >= 0)
    Then WriteLn((2 + x)/(x * x - 9) + SQRT(x + 3))
    Else WriteLn('вираз не має змісту');
    ReadLn
End.
```

Зверніть увагу, отримаємо результат у вигляді з рухомою крапкою.

Тести до задачі:

Значення x	Результат
1	$\frac{2+1}{1^2-9}+\sqrt{1+3}=0{,}375+2=1{,}625$
3	вираз не має змісту
-10	вираз не має змісту

Складений оператор

Якщо після слів Then або Else треба написати два або більше операторів, то їх треба брати в *операторні дужки* Begin і End.

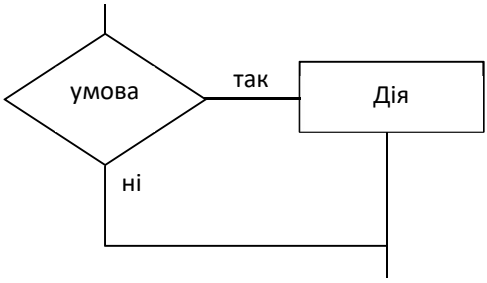
Розв’язок задачі 5 може бути записаний дещо по-іншому:

```
Var x y: Real;
Begin
    ReadLn(x);
    If (x * x - 9 <> 0) And (x + 3 >= 0) Then
    Begin
        y := (2 + x) / (x * x - 9) + SQRT(x + 3);
        WriteLn(y:8:4);
    End
    Else WriteLn('вираз немає змісту');
    ReadLn
End.
```

У цій програмі результат буде у вигляді з фіксованою крапкою – десятковий дріб з 4-а знаками після коми завдяки формату :8:4.

Коротка форма розгалуження

Це конструкція, яка має такий вигляд:



Якщо умова істинна, виконати дію, якщо умова хибна, не робити нічого. Мовою Паскаль:

```
If умова Then дія;
```

Задача 1. Дано три цілих числа. Знайти найбільше з них.

Розв'язок. Позначимо ці числа a, b, c.

```
Var a, b, c, y: Integer;
Begin
    ReadLn (a, b, c);
    If (a >= b) And (a >= c) Then y := a;
    If (b >= a) And (b >= c) Then y := b;
    If (c >= a) And (c >= b) Then y := c;
    WriteLn(y);
    ReadLn
End.
```

Задача 2. Дано число n, яке позначає бал, який виставляють у школі. Вивести, до якого рівня належить введена оцінка.

```
Var n: Byte;
Begin
    ReadLn(n);
    If (n = 1) Or (n = 2) Or (n = 3) Then WriteLn('I рівень');
    If (n = 4) Or (n = 5) Or (n = 6) Then WriteLn('II рівень');
    If (n = 7) Or (n = 8) Or (n = 9) Then WriteLn('III рівень');
    If (n = 10) Or (n = 11) Or (n = 12) Then WriteLn('IV рівень');
    ReadLn
End.
```

Задачі з Алготестера

Задачі з Алготестера на розгалуження:

«Апельсини» (0481),

«Цікава гра» (0181),

«Спекотні дні пінгвінів» (0163).

Ці задачі можна рекомендувати як групові проекти на 1-2 заняття.

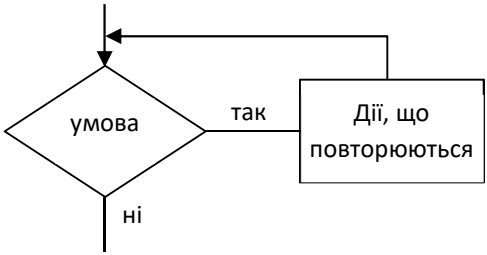
Висновок

Програми з розгалуженням містять оператори If. Під час виконання цих програм завжди є оператори, які для певних вхідних даних не виконуються. Тобто, комп'ютер завдяки умовам робить вибір, що виконувати. Оператори розгалуження змушують комп'ютер «думати».

Розділ ІІІ. Програми з циклами

Цикл – це багаторазове повторення фрагменту програми. Цикли (або повторення) дають змогу записати дії один раз, а виконувати їх комп'ютер може багато разів.

У блок-схемах структура циклу має вигляд



У мовах програмування є декілька видів операторів циклу. Розглянемо два із них:

1) оператор циклу з передумовою;
2) оператор циклу з параметром.

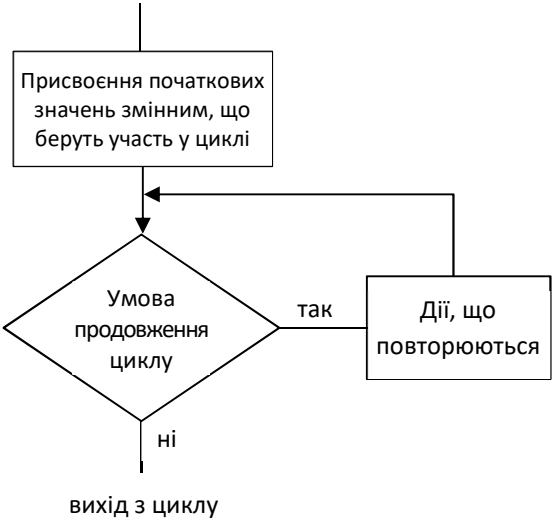
Оператор циклу з передумовою

На Паскалі оператор циклу з передумовою має вигляд

```
While умова Do дії;
```

Читаємо: «поки умова істинна, виконувати дії».

Блок-схема оператора циклу з передумовою має вигляд:



Оператор виконується так:

1) комп'ютер перевіряє умову;
2) якщо умова істинна, виконує дії;
3) якщо умова хибна, то вихід з циклу, тобто кінець циклу;
4) повернутися на пункт 1.

Розглянемо виконання цього оператора на прикладах.

Вправа 1. Виконати послідовність операторів:

```
i := 1;
While i <= 4 Do
Begin
    WriteLn(i);
    i := i + 1
End;
```

Покрокове виконання можна показати у такій таблиці:

Значення i	Умова i ≤ 4	На екрані
1	1 ≤ 4 (істина)	1
2	2 ≤ 4 (істина)	2
3	3 ≤ 4 (істина)	3
4	4 ≤ 4 (істина)	4
5	5 ≤ 4 (хиба)	

Отже, після виконання команд отримаємо числа 1, 2, 3, 4 виведені на екран у стовпець.

Вправа 2. Змінити команди з вправи 1 так, щоб отримати на екрані

а) числа у рядок з одним пропуском між ними;

б) числа від 1 до 100 в один рядок;

в) тільки непарні числа менші від 100;

г) тільки парні числа менші від 100;

Відповіді:

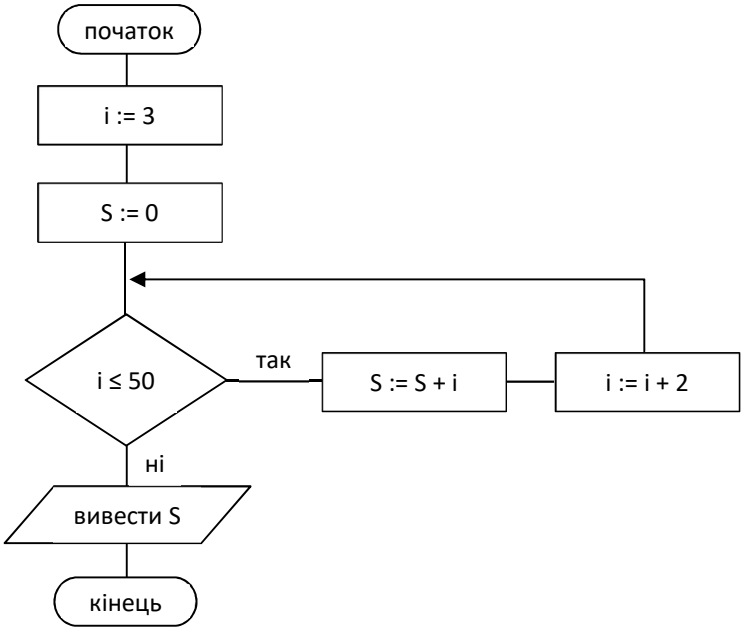
а) замість WriteLn(i) написати Write(i, ' ');

б) умова буде i <= 100;

в) умова буде i <= 100, замість i := i + 1 треба i := i + 2

г) все як в) і замість i := 1 записати i := 2.

Вправа 3. Записати програму алгоритму, зображеного на блок-схемі:



Аналіз блок-схеми. Змінна i – це параметр циклу, вона «керує» циклом. Значення i будуть 3, 5, 7, … 51, отже тип змінної i: Byte. Змінна S отримає значення 0, 3, 8, 15, 24… і позначає в алгоритмі суму значень i. Ця сума, можливо, буде більшою за 256. Отже, тип S: Word.

Програма має вигляд:

```
Var i: Byte;
    S: Word;
Begin
    i := 3;
    S := 0;
    While i <= 50 Do
    Begin
        S := S + i;
        i := i + 2;
    End;
    WriteLn(S);
    ReadLn
End.
```

Вправа 4. Скільки разів виконається тіло циклу (записується після Do), якщо

а) перед циклом буде замість i := 3 такий оператор:

- i := 50;
- i := 51;

Відповідь:

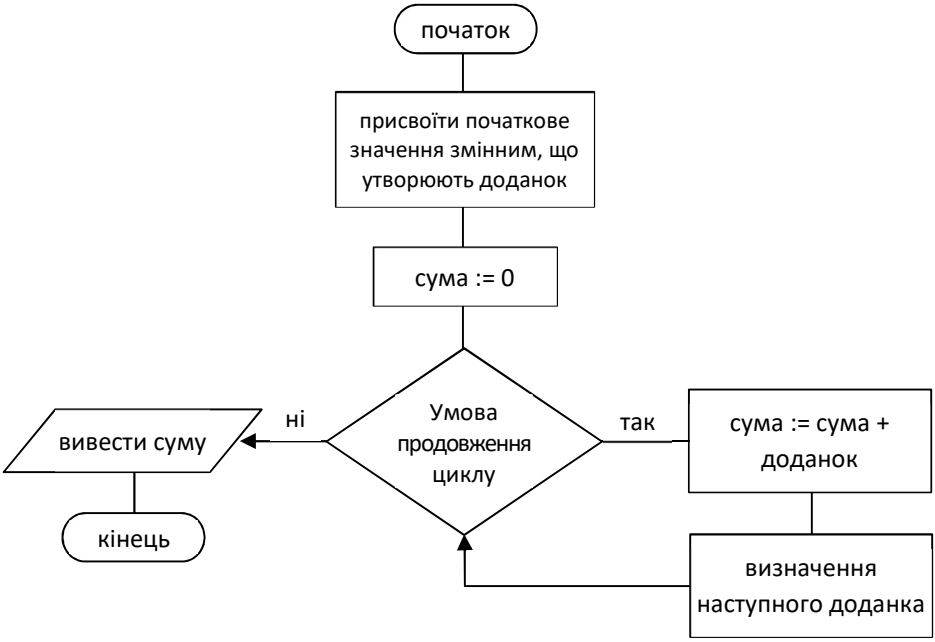
- один раз;
- жодного разу.

б) у циклі забрати команду i := i + 2

Відповідь: значення i буде залишатися незмінним, тобто i = 3 і умова i <= 50 буде істинна завжди, тобто цикл буде виконуватися без кінця. Така ситуація має назву *зациклення*. Зазвичай це помилка, яка призводить до зависання програми.

Задачі з циклом While

Типовою є задача знаходження суми доданків, які обчислюють за певною формулою, або вводять з клавіатури. Блок-схема задачі має вигляд:

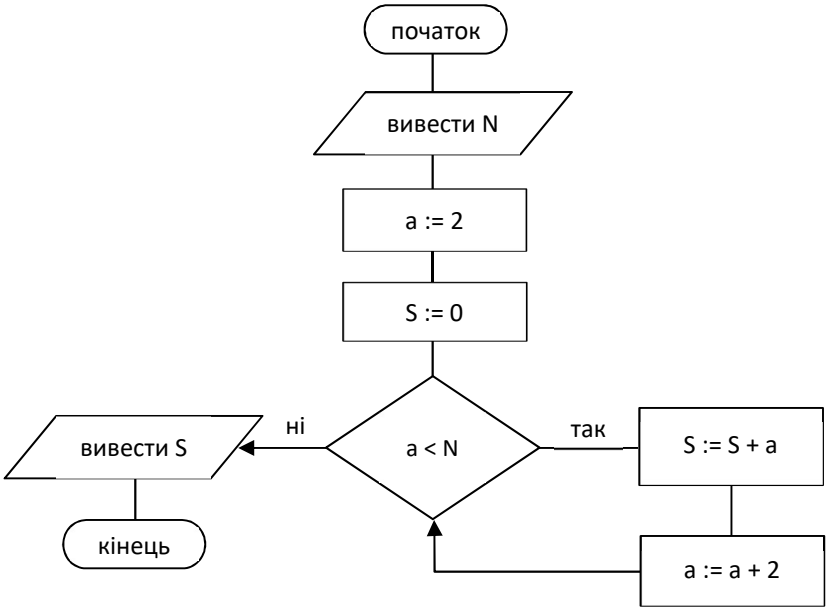


Розглянемо конкретні задачі.

Задача 1. Знайти суму парних чисел, менших за число N.

Розв'язання. Позначимо S – сума, a – доданок. Змінна a буде мати значення спочатку 2, після нього 4, далі 6, 8, … Останнім числом буде N - 1, якщо N непарне число, або N - 2, якщо N – парне число. Але перевіряти N на парність немає потреби. Умова у циклі a < N забезпечить останній доданок.

Блок-схема розв’язку задачі:



Програма на Паскалі:

```
Var N, S, a: Word;
Begin
    ReadLn(N);
    a := 2;
    S := 0;
    While a < N Do
    Begin
        S := S + a;
        a := a + 2;
    End;
    WriteLn(S);
    ReadLn
End.
```

Складемо тести для програми:

N	S
2	0
3	2
4	2
5	0 + 2 + 4 = 6
6	0 + 2 + 4 = 6
15	0 + 2 + 4 + 6 + 8 + 10 + 12 + 14 = 56

Задача 2. Знайти суму непарних чисел, що не перевищують число N.

Розв'язання. Замініть на a := 1, умова циклу буде a <= N.

Задача 3. Знайти суму

$$\frac{1}{2}+\frac{2}{3}+\frac{3}{4}+\cdots+\frac{N}{N+1}$$

Розв'язання. Позначимо a – чисельник доданка, b – знаменник доданка.

```
Var a, b, N: Word;
    S: Real;
Begin
    ReadLn(N);
    a := 1;
    b := 2;
    S := 0;
    While a <= N Do
    Begin
        S := S + a / b;
        a := a + 1;
        b := b + 1;
```

```
    End;
    WriteLn(S:8:4);
End.
```

Пригадаємо, S:8:4 означає, що число S буде виведене на екран у вигляді з фіксованою крапкою з чотирма десятковими розрядами після неї.

Задача 4. Знайти суму довільних цілих N чисел, які ввести з клавіатури.

Розв'язання. Нехай a – число яке вводять з клавіатури і воно одночасно є доданком.

Введемо ще одну змінну i, яка буде рахувати кількість введених доданків:

```
Var a, s: Integer;
    i, N: Word;
Begin
    ReadLn(N);
    S := 0;
    i := 1;
    While i <= N Do
    Begin
        ReadLn(a);
        S := S + a;
        i := i + 1;
    End;
    WriteLn(S);
    ReadLn
End.
```

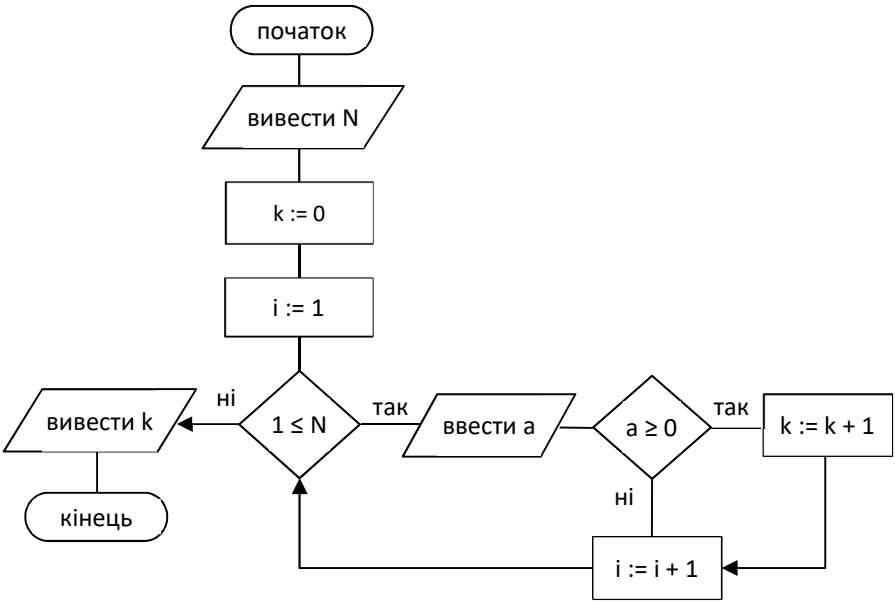
В Алготестері задача «Марічка і печиво» 0011 саме і є задачею на знаходження суми у циклі.

Ще одним типом задач на While є знаходження кількості чисел, які задовольняють певну умову. Розглянемо деякі з них.

Задача 5. Серед введених з клавіатури N цілих чисел знайти, скільки є додатних.

Розв'язання. Позначимо a – число, що вводять з клавіатури, i – номер цього числа, k – кількість додатних чисел. Пригадаємо, що умова додатного числа a > 0.

Блок-схема алгоритму розв'язування задачі:



Програма:

```
Var N, k, s: Word;
    a: Integer;
Begin
    ReadLn(N);
    k := 0;
    i := 1;
```

```
    While i <= N Do
    Begin
        ReadLn(a);
        If a > 0 Then k := k + 1;
        i := i + 1;
    End;
    WriteLn(k);
    ReadLn
End.
```

Оператор циклу з параметром

Досить часто трапляються цикли, у яких крок зміни змінної, яка входить в умову, дорівнює одиниці. У таких випадках можна використати *цикл з параметром*, який записується у коротшій формі, порівнюючи з циклом з передумовою. Отже, таку конструкцію:

```
i: = iпоч;
While i <= iкін Do
Begin
    тіло циклу
    i := i + 1;
End;
```

можна записати циклом з параметром

```
For i := iпоч To iкін Do тіло циклу;
```

Читаємо «Для і від $i_{поч.}$ до $i_{кін}$ з кроком 1 виконати тіло циклу». Оператор виконується так:

1) i := $i_{поч}$;
2) перевірка умови i <= $i_{кін}$;

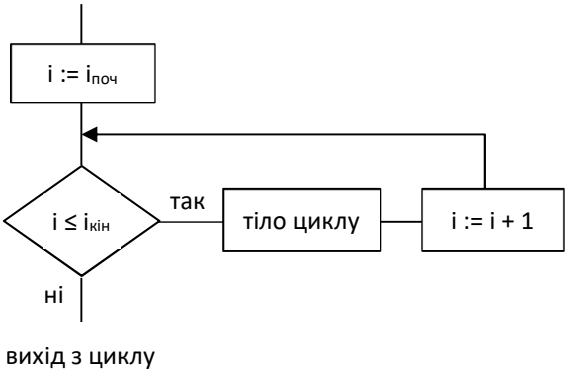
3) якщо умова хибна, то кінець оператора;

4) якщо умова істинна, то виконується тіло циклу;

5) i := i + 1;

6) повернення на пункт 2)

Змінні i, $i_{поч.}$, $i_{кін}$ обов'язково цілого типу. Графічно цикл має такий вигляд:



Вправа 1. Виконати оператор

```
For i := 1 To 3 Do WriteLn(i);
```

i	Умова $i \leq 3$	Значення умови	На екрані
1	$1 \leq 3$	істина	1
2	$2 \leq 3$	істина	2
3	$3 \leq 3$	істина	3
4	$4 \leq 3$	хиба	

Вправа 2. Що треба змінити в операторі з вправи 1, щоб отримати на екрані числа 10; 11; 12; …100?

Відповідь: `For i := 10 To 100`

Вправа 3. Виконати послідовність операторів

```
S := 0;
a := 1;
For k := 2 To 4 Do
Begin
    S := S + a * k;
    a := -a;
End.
```

Розв'язок:

S	a	k	Умова k ≤ 4
0	1		
		2	2 ≤ 4 істина
0 + 1 * 2 = 2	-1	3	3 ≤ 4 істина
2 + (-1) * 3 = -1	1	4	4 ≤ 4 істина
-1 + 1 * 4 = 3	-1	5	5 ≤ 4 хиба

Задача 1. Знайти середнє арифметичне N цілих чисел, які ввести з клавіатури.

Розв'язання. Позначимо число, яке вводять, буквою a. Для знаходження середнього арифметичного треба знайти суму S цих N чисел, потім поділити S на N. Змінні a, S, N цілого типу, а вираз S/N дійсного типу, тому для виведення середнього арифметичного задамо формат виведення :8:4, тобто 4 знаки після коми у десятковому дробі. Програма має вигляд:

```
Var i, a, S, N: Integer;
Begin
    ReadLn(N); {вводимо кількість чисел}
    S := 0;    {задаємо початкове значення для суми}
    For i := 1 To N Do
    Begin
```

```
        ReadLn(a);    {N разів вводимо a}
        S := S + a;   {збільшуємо суму на a}
    End;
    WriteLn(S/N:8:4); {виводимо середнє арифметичне}
    ReadLn
End.
```

✱ Тут у фігурних дужках {…} записаний коментар, який ніяк не впливає на виконання програми. Коментар пояснює або документує програму.

Складаємо тест для перевірки цієї програми. Нехай N = 5; а числа, які вводять 3; 7; 19; 11; 40. Тоді результат буде

(3 + 7 + 19 + 11 + 40) : 5 = 80 : 5 = 16.0000

Задача з Алготестера. В Алготестері є задача 0011 «Марічка і печиво», яку можна розв'язати, використовуючи цикл з параметром. Математична модель задачі зводиться до знаходження суми N чисел, що вводять з клавіатури. Вкінці від знайденої суми треба відняти кількість всіх чисел тобто N.

Особливістю цієї задачі є те, що коли додавати дуже багато чисел типу Longint, сума може бути настільки великою, що не поміщається у тип Longint, тому якщо позначити суму S, то опис буде S: Real; а щоб вивести S як ціле число, треба задати формат виведення S:20:0.

Програма може мати вигляд:

```
Var S: Real;
    a, N, i: Longint;
Begin
    ReadLn(N);
    S := 0;
    For i := 1 To N Do
```

```
    Begin
        ReadLn(a);
        S := S + a;
    End;
    S := S - N;
    WriteLn(S:20:0);
    ReadLn
End.
```

Розв'язування задач з циклами

Задача 1. Алгоритм Евкліда. Знайти найбільший спільний дільник двох натуральних чисел.

Пригадаємо, що НСД (a, b) – це таке найбільше число, на яке числа a, b діляться без остачі. Евклід придумав такий спосіб знаходження НСД (a, b): поки числа різні (тобто a ≠ b) замінювати більше число різницею більшого та меншого чисел.

Розв'язання. Тут треба застосувати цикл із передумовою, оскільки наперед невідомо, скільки буде повторень:

```
Var a, b: Word;
Begin
    ReadLn(a, b);
    While a <> b Do
        If a > b Then a := a - b Else b := b - a;
    WriteLn(a);
    ReadLn
End.
```

Задача 2. Утворити і вивести на екран всі числа Фібоначчі, менші за 1000. Послідовністю Фібоначчі називають числа 1, 1, 2, 3, 5, 8, 13, 21, …, кожне наступне число дорівнює сумі двох попередніх чисел.

Розв'язання. Оскільки числа мають бути меншими, ніж 1000, то ми не знаємо наперед скільки їх буде. Треба використати цикл з передумовою. Позначимо числа послідовності буквами a, b, c; а пізніше будемо посуватися на одне число далі a := b; b := c;

Отже, програма:

```
Var a, b, c: Word;
Begin
    a := 1; b := 1;
    Write(a);
    While b < 1000 Do
    Begin
        Write(' ', b);
        c := a + b;
        a := b;
        b := c;
    End;
    ReadLn
End.
```

Задача 3. Серед двоцифрових чисел знайти кількість таких, що у своєму записі містять цифру а.

Аналіз задачі. Позначимо число буквою x, виділити цифри із числа x можна діями x div 10 - неповна частка – кількість десятків числа x; x mod 10 – залишок від ділення на 10 – кількість одиниць числа x. Наприклад, якщо x = 23, то 23 div 10 = 2 і 23 mod 10 = 3. Часто плутають, що таке цифра і що таке число, тому варто нагадати, що цифра – це символи 0, 1, …, 9 для запису чисел. Оскільки тут відомо, що x = 1, 2, 3, …, 99, то можна використати цикл з параметром:

```
Var x, a, k: Byte;
Begin
    ReadLn(a);
    k := 0;
```

```
    For x := 1 To 99 Do
        If (x mod 10 = a) Or (x div 10 = a)
        Then k := k + 1;
    WriteLn(k);
    ReadLn
End.
```

Тут змінною k позначено лічильник. Оскільки спочатку нема жодного числа, що задовольняє умову задачі, то k надаємо початкове значення 0, а далі у циклі, коли знайшли шукану цифру a у числі x, збільшуємо лічильник на 1 оператором k := k + 1.

Тестом для задачі можна задати a = 0, тоді є 9 чисел, що містять цифру нуль, це 10, 20, 30, 40, 50, 60, 70, 80, 90.

Задача з Алготестера 0121 «Літня школа». Треба довільним чином утворити K команд із N людей. У кожній команді може бути 1, 2 або 3 людини. Вивести слово «Impossible», якщо це зробити не є можливо, або кількість людей у командах в один рядок з одним пропуском між ними.

Аналіз задачі. Розглянемо всі можливі випадки.

1) Коли K > N, то не знайдеться людей на всі команди; коли 3K < N, то людей буде більше, ніж треба на K команд. Отже, треба вивести 'Impossible':

```
If (K > N) OR (3 * K < N) Then WriteLn('Impossible');
```

2) Коли K = N, то всі команди будуть по одній людині:

```
If K = N Then For i := 1 to K Do Write(1, '␣');
```

3) Коли 2K = N, то всі команди будуть по два учасники:

```
If 2 * K = N Then For i := 1 to K Do Write(2, '␣');
```

4) Коли 3K = N, то всі команди будуть по 3 учасники:

```
If 3 * K = N Then For i := 1 to K Do Write(3, '␣');
```

5) Коли K < N < 2K, тоді буде 2K - N команд по одному учаснику і N - K по два учасники:

```
If (K < N) AND (N < 2 * K) Then
Begin
    For i := 1 To 2 * K - N Do Write(1, '␣');
    For i := 1 To N - K Do Write(2, '␣');
End;
```

6) Коли 2K < N < 3K, тоді буде 3K - N команд по дві людини і N - 2K команд по троє людей:

```
If (2 * K < N) And (N < 3 * K) Then
Begin
    For i := 1 To 3 * K - N Do Write(2, '␣');
    For i := 1 To N - 2 * K Do Write(3, '␣');
End.
```

Тепер треба всі ці оператори зібрати у програму, додавши на початку

```
Var K, N, i: Byte;
Begin
    ReadLn(N, K);
```

а в кінці дописати

```
    ReadLn;
    ReadLn
End.
```

Задачі з Алготестера

Для роботи з циклами можна запропонувати такі задачі:
«Марічка і печиво» (0011),
«Літня школа» (0121),
«Непередбачувана погода» (1301),
«Перехід дороги» (1401),
«Щасливенькі числа» (1402),
«Санта Клаус і сума цифр» (1539).

Висновок

Оператори циклу дають змогу записати дії, які комп’ютер виконує багато разів. Компактна, невелика за розмірами програма може задавати тривалий процес опрацювання даних.

Розділ IV. Табличні величини

У Паскалі табличні величини називають масивами. Розрізняють одно та двовимірні масиви.

Якщо є багато величин одного типу, над якими у програмі треба виконати одинакові дії, тоді є необхідність використовувати структуровані дані, до яких належать масиви.

Лінійні масиви

Масив – це набір величин однакового типу. Величини у масиві називають *елементами*. Кожний елемент має своє місце у масиві, яке визначається його порядковим номером – *індексом*. Кожен масив має свою назву.

Назвемо масив буквою А. Якщо у масиві є чотири елементи, то їх можна записати А[1], А [2], А[3], А[4]. А[1] читаємо «А перше», або «А з номером 1», або «А з індексом 1».

Наприклад, якщо у цьому масиві записані оцінки школяра, то елементи можуть мати такі значення А[1] = 10; А[2] = 9; А[3] = 11; А[4] = 10.

Нехай масив А має N елементів, тоді загальний вигляд цього масиву буде А[1], А [2], А[3], …, А [N].

Елемент масиву позначають А[i] (читаємо А і-те, або А з індексом і), де змінна і набуває по черзі значень 1, 2, 3, …, N. Оскільки і – це порядковий номер елемента, то ця змінна є цілого типу.

Якщо у програмі треба виконати однакові дії над усіма елементами, то використовують цикл:

```
For i := 1 To N Do дії над A[i];
```

Опис масиву

Опис масиву має вигляд

```
Var ім’я масиву: Array [номер першого елемента..номер останнього елемента] Of тип елементів;
```

Наприклад:

1. `Var A: Array [1..4] Of Byte;`

описує масив A, що має 4 елементи типу Byte. Комп’ютер виділить 4 * 1 = 4 байти пам’яті для елементів масиву.

2. `Var B: Array [1..366] Of Integer;`

описує масив B, що має 366 елементів типу Integer. І у пам’яті комп’ютера буде виділено 366 * 2 байти пам’яті.

Введення масиву

Розглянемо масив A, що має N елементів, тобто A[1..N].

Якщо значення елементів масиву треба ввести з клавіатури, можна використати один з двох варіантів:

1. Якщо значення вводять через пропуск в одному рядку:

```
For i := 1 To N Do Read(A[i]);
```

наприклад:

```
For i := 1 To 4 Do Read(A[i]);
```

Під час виконання оператор вимагає ввести чотири числа в один рядок через пропуск і після останнього числа натиснути Enter. Це може мати такий вигляд:

```
10␣9␣11␣10 Enter
```

2. Якщо значення вводять по одному числу у рядку:

```
 For i := 1 To 4 Do ReadLn(A[i]);
```

наприклад:

```
For i := 1 To 4 Do ReadLn(A[i]);
```

Оператор вимагає ввести чотири числа, після кожного – Enter. Це може мати такий вигляд:

```
10 Enter
9 Enter
11 Enter
10 Enter
```

В обох випадках будемо мати

```
A[1] = 10;
A[2] = 9;
A[3] = 11;
A[4] = 10.
```

Виведення масиву

Якщо значення масиву A[1..N] треба вивести на екран монітора, можна використати один з двох варіантів:

1. Якщо значення виводять у рядок з одним пропуском між ними:

```
For i := 1 To N Do Write(A[i], '␣');
```

2. Якщо значення виводять у стовпець – по одному значенню у кожному рядку:

```
For i := 1 To N Do WriteLn(A[i]);
```

Утворення масиву за формулою

Задати значення елементів масиву можна за формулою. Утворення елементів масиву A[1..N] за формулою має вигляд:

```
For i := 1 To N Do A[i] := формула;
```

Розглянемо приклади. Утворити масиви, елементи яких

1) дорівнюють своєму індексу. Якщо елемент A[i], то його індекс i, тобто A[i] := i:

```
For i := 1 To N Do A[i] := i;
```

2) вдвічі більші за свої індекси:

```
For i := 1 To N Do A[i] := i * 2;
```

3) утворені за формулою $A[i] = i^2 - i$:

```
For i := 1 To N Do A[i] := i * i - i;
```

4) всі дорівнюють числу 41:

```
For i := 1 To N Do A[i] := 41;
```

Задачі на використання масивів

Задача 1. Знайти суму елементів масиву зі ста цілих чисел, введених з клавіатури.

Позначимо масив буквою A, i – номер елемента, S – сума. Програма матиме вигляд:

```
Var A: Array[1..100] Of Integer;
    i: Byte;
    S: LongInt;
Begin
    For i := 1 To 100 Do Read(A[i]);
    S := 0;
    For i := 1 To 100 Do S := S + A[i];
    WriteLn(S);
    ReadLn;
    ReadLn
End.
```

Можна записати виведення і обчислення суми в одному циклі. Тоді програма буде мати такий вигляд:

```
Var A: Array[1..100] Of Integer;
    i: Byte;
    S: LongInt;
Begin
    S := 0;
    For i := 1 To 100 Do
    Begin
```

```
        Read(A[i]);
        S := S + A[i];
    End;
    WriteLn(S);
    ReadLn;
    ReadLn
End.
```

Змінну S описали, як Longint тому, що якщо додавати 100 чисел типу Integer, результат можна отримати такий, що виходить за межі цього типу.

Задача 2. Елементи масиву B[1..200] утворені, як синус свого індексу. Знайти кількість від'ємних елементів.

Аналіз задачі. Згідно з умовою, B[i] = sin i, отже тип елементів – Real. Позначимо k – кількість від'ємних елементів, B[i] < 0. Оскільки, функція sin x є періодична з періодом 2π, очікуваний результат для k буде приблизно 100.

Програма матиме вигляд:

```
Var B: Array[1..200] Of Real;
    i, k: Byte;
Begin
    k := 0;
    For i := 1 To 200 Do
    Begin
        B[i] := sin(i);
        if B[i] < 0 Then k := k + 1;
    End;
    WriteLn(k);
    ReadLn
End.
```

Задача 3. Масив цілих чисел M[1..N] ввести з клавіатури в один рядок. Дати відповідь на запитання чи є у цьому масиві число A. Обмеження: 1 < N < 1000.

Аналіз задачі. На відміну від попередніх задач, у масиві M невідома кількість елементів, оскільки N може набувати значень від 2 до 999. У таких випадках треба описати масив максимально можливої довжини, тобто M[1..999].

Додамо до масиву M[N + 1] = A. Підправимо в описі розмір масиву на 1 більший, тобто M[1..1000]. Використаємо цикл з передумовою для пошуку числа A у масиві:

```
i := 1; While M[i] <> A Do i := i + 1;
```

Якщо у масиві M є число A, то будемо мати значення i ≤ N. Якщо у масиві M нема числа A, будемо мати завершення циклу на значенні i = N + 1, оскільки M[N + 1] = A.

Далі перевіримо, якщо i ≤ N дамо відповідь 'Yes' – є число A в масиві M, якщо ні (тобто i = N + 1) дамо відповідь 'No' – нема такого числа.

Програма:

```
Var M: Array[1..1000] Of Integer;
    A, i, N: Integer;
Begin
    ReadLn(N);
    For i := 1 To N Do Read(M[i]);
    ReadLn (A);
    M[N + 1] := A;
    i := 1;
    While M[i] <> A Do i := i + 1;
    If i <= N Then WriteLn('Yes') Else WriteLn('No');
    ReadLn;
    ReadLn
End.
```

Тест. Введемо N = 4, елементи масиву 5; 7; 9; 18 і число A:

- якщо A = 9 – отримаємо Yes;
- якщо A = 20 – отримаємо No.

Задачі з Алготестера

Для роботи з одновимірними масивами можна розв'язати такі задачі:

«Борщ, картопля і салат» (0012),

«Верховна Рада» (0022),

«Коля, Вася і лабіринт» (0032),

«Тренер слонів» (0114),

«Зуби» (0182),

«Вогняне дихання» (0183),

«Марічка і смачні цукерки» (0361),

«Помічники Санта Клауса» (1531).

Двовимірні масиви

Прямокутні таблиці, які містять дані одного типу можна описати у Паскалі як *двовимірні масиви*. Елемент двовимірного масиву має два індекси – номер рядка і номер стовпця, на перетині яких він розташований.

Розглянемо, який вигляд має масив A, що має N рядків і M стовпців:

```
A[1, 1]; A[1, 2]; A[1, 3]; ...; A[1, M];
A[2, 1]; A[2, 2]; A[2, 3]; ...; A[2, M];
A[3, 1]; A[3, 2]; A[3, 3]; ...; A[3, M];
.....    .....    .....    .....
A[N, 1]; A[N, 2]; A[N, 3]; ...; A[N, M].
```

Всіх елементів є N * M.

Прикладом двовимірного масиву є таблиця множення:

```
1  2  3 ...  9
2  4  6 ... 18
3  6  9 ... 27
...  ...  ...
9 18 27 ... 81
```

Елемент масиву A можна записати, як A[i, k], де i послідовно набуває значень 1, 2, …, N, а змінна k набуває значень 1, 2, …, M.

Опис масиву A, що має 100 рядків і 50 стовпців має вигляд:

```
Var A: Array[1..100, 1..50] Of тип елементів;
```

Опрацювання всіх елементів за одним правилом можна записати подвійним (вкладеним) циклом:

```
For i := 1 To N Do
    For k := 1 To M Do
        Дія з A[i, k];
```

Приклади простих задач із двовимірними масивами

Задача 1. Таблиця множення. Утворити масив, що має 9 рядків і 9 стовпців, елементи якого дорівнюють добутку своїх індексів.

Розв'язок:

```
Var i, k: Byte;
    A: Array [1..9, 1..9] Of Byte;
Begin
    For i := 1 To 9 Do
    Begin
```

```
        For k := 1 To 9 Do
        Begin
            A[I, k] := i * k;
            Write(A[i, k]:5);
        End;
        WriteLn; {перейти на новий рядок}
    End;
    ReadLn
End.
```

Задача 2. Елементи двовимірного масиву утворені як сума своїх індексів. Знайти суму елементів цього масиву. Масив має не більше 15 рядків та 15 стовпців.

Розв'язання. Позначимо кількість рядків і стовпців у масиві відповідно n і m, ці числа можуть набувати значень від 1 до 15, тобто реально масив може мати менші розміри, залежно від вхідних даних. В цьому разі треба описувати масив максимально великих розмірів, передбачених умовою задачі:

```
Var i, k, n, m: Byte;
    s: Word;
    A: Array [1..15, 1..15] Of Byte;
Begin
    ReadLn(n, m);
    S := 0;
    For i := 1 To n Do
    Begin
        For k := 1 To m Do
        Begin
            A[i, k] := i + k;
            s := s + A[i, k];
        End;
    End;
    WriteLn(s);
```

```
    ReadLn
End.
```

Задача 3. Увести масив цілих чисел в межах від 1 до 1000 розміром n x m. Знайти найменший елемент та його розташування.

Позначимо масив – A[1..n, 1..m], найменший елемент – Min, номер рядка, де розташований цей елемент – x, номер стовпця – p. Спочатку найменшим будемо вважати перший елемент, тобто Min = A[1, 1]. Далі переглядаємо послідовно всі елементи, якщо знайдемо менший, запам'ятовуємо його як найменший. Оскільки розміри масиву невідомі, описати треба максимально можливі:

```
Var i, k, n, m, p, x, Min: Word;
    A: Array [1..1000, 1..1000] Of Word;
Begin
    ReadLn(n, m);
    For i := 1 To n Do
        For k := 1 To m Do
            ReadLn(A[i, k]);
    Min := A[1, 1];
    x := 1;
    p := 1;
    For i := 1 To n Do
        For k := 1 To m Do If A[i, k] < Min Then
        Begin
            Min := A[i, k];
            x := i;
            p := k
        End;
    WriteLn(Min);
    WriteLn(x, p);
    ReadLn
End.
```

Задачі з Алготестера

Задачі для роботи з двовимірними масивами:

«Де вони?» (0014),

«Прямокутна країна» (0051),

«Лотерея» (0061),

«Цікаве листування» (0151).

Висновки

Масиви дають змогу позначати однією буквою велику кількість однотипних величин, а також опрацьовувати їх у циклах. Це робить запис програм опрацювання великої кількості однотипних величин компактним.

Розділ V. Дані рядкового типу

Комп'ютер опрацьовує дані різних типів, в тому числі і тексти. Послідовності будь-яких символів називають *рядками* або *даними рядкового типу*. У Паскалі значення рядків беруть у одинарні лапки (апострофи).

Приклади рядків:

```
'Word';
'Hello';
'Spring';
'a + b = ';
'Year 2020'.
```

У Паскалі є декілька типів рядкових величин, одним із яких є тип String (про інші типи можна прочитати в інтернеті). Тип String займає у пам'яті комп'ютера 256 байтів, його значення може вмістити 255 різних символів. Кожен символ займає 1 байт, а у байті з номером 0 зберігається кількість символів рядка.

Опис рядкової величини має вигляд:

```
Var змінна: String;
```

Виділяється 256 байтів для вказаної змінної.

Щоб оголосити рядок меншої довжини, можна явно вказувати кількість символів у квадратних дужках. Наприклад:

```
Var a: String [50];
    b: String [10];
```

Для змінної a буде виділено 51 байт (50 байтів для символів рядка і додатковий байт з номером 0 для зберігання довжини рядка), а для змінної b – 11 байтів.

Операції над рядками

Рядки можна додавати (об’єднувати):

```
'I' + ' Love' + ' You' = 'I Love You'
```

Зрозуміло, що для рядкових величин a + b і b + a дають різні результати.

Рядки можна порівнювати операторами порівняння =, <>, >, <, <=, >=. Порівнюються коди символів.

До символів у рядку можна звернутися як до елементів лінійного масиву. Якщо описати змінну A, як рядкову величину завдовжки 5 символів

```
Var A: String[5];
```

і присвоїти A значення 'Hello':

```
A = 'Hello'
```

тоді:

```
A[1] = 'H';
A[2] = 'e';
A[3] = 'l';
A[4] = 'l';
A[5] = 'o'.
```

Функції та процедури для рядків

Для даних рядкового типу string є багато функцій та процедур, опис яких можна знайти в інтернеті. Розглянемо деякі з них.

Введемо такі функції:

1) `Length(рядок)` визначає кількість символів у рядку:

```
Length('Hello') = 5
```

2) `Copy(рядок, p, m)` повертає m символів з рядка, починаючи від символу з номером p:

```
Copy('Hello, Spring', 8, 6) = 'Spring'
```

3) `Pos(p1, p2)` визначає номер символу, з якого починається входження рядка p1 у рядок p2:

```
Pos ('Spring', 'I_Love_Spring') = 8.
```

Розглянемо деякі процедури:

1) `Str(число, рядок)` перетворює число у рядок:

```
Str(123456, x);
```

Після цього x = '123456'.

2) `Val (рядок, число, ознака)` перетворює рядкове дане у число. Рядок має бути коректним записом числа, тобто складатися лише із цифр. Якщо це не так, число = 0, ознака <> 0. У разі вдалого перетворення ознака = 0.

Наприклад:

```
Val('123456', x, c);
```

Отримаємо x = 123456, c = 0.

```
Val('1e23456', x, c);
```

Отримаємо x = 0, c = 2, тобто перетворення є не вдалим.

Прості задачі із рядками

Задача 1. Скільки у введеному з клавіатури тексті є букв 'a'?

Розв'язання. Позначимо

t – текст,
k – кількість букв 'a',
i – номер букви у тексті.

Будемо брати з рядка по одній букві і порівнювати з 'a'.

Програма:

```
Var t: String;
    k, i: Byte;
Begin
    ReadLn(t);
    k := 0;
    For i := 1 To Length(t) Do
        If t[i] = 'a' Then k := k + 1;
    WriteLn(k);
    ReadLn
End.
```

Задача 2. Скільки разів у введеному з клавіатури тексті є слів 'And'?

```
Var t: String;
    k, i: Byte;
Begin
    ReadLn(t);
    k := 0;
    For i := 1 To Length (t) Do
        If Copy(t, i, 3) = 'And' Then k := k + 1;
    WriteLn(k);
    ReadLn
End.
```

Задача 3. Дати відповідь чи є у введеному з клавіатури числі цифра 5.

Розв'язання. Позначимо x – дане число, перетворимо його у рядок a, кожний символ рядка a порівняємо з символом '5'.

Програма:

```
Var X: Longint;
    a: String[12];
    i: Byte;
Begin
    ReadLn(x);
    Str(x, a);
    i := 1;
    While(a[i] <> '5') And (i <= Length(a)) Do
        i := i + 1;
    If i <= Length (a)
    Then WriteLn('Yes')
    Else WriteLn('No');
    ReadLn
End.
```

Задача 4. Вивести на екран введений з клавіатури текст у зворотному порядку.

Розв'язання. Позначимо a – введений текст, b – результат, i – номер букви у тексті a. Будемо вибирати по одній букві із тексту a і додавати до тексту b. Спочатку у тексті b нема жодного символу, b := '' (пустий рядок);

Програма:

```
Var a, b: String;
    i: Byte;
Begin
    ReadLn(a);
    b := '';
    For i := 1 To Length(a) Do b := a[i] + b;
    WriteLn(b);
    ReadLn
End.
```

Задача 5. Визначити кількість речень у введеному з клавіатури тексті. Практично треба знайти кількість таких символів як .?!

Програма:

```
Var a: String;
    k, i: Byte;
Begin
    ReadLn(a);
    k := 0;
    For i := 1 To Length(a) Do
        If (a[i] = '.') Or (a[i] = '?') Or (a[i] = '!')
        Then k := k + 1;
    WriteLn(k);
ReadLn
End.
```

Задача 6. Дати відповідь, чи можна із букв слова A скласти слово B.

Розв'язання. Будемо вважати спочатку, що із букв слова A можна скласти слово B, тобто R := 'Yes'. Надалі будемо кожну букву слова B шукати у слові A. Якщо таку букву знайдено, то будемо її замінювати на пропуск у слові A. Якщо деяку букву не знайдено у слові A, то виконуємо R := 'No'.

```
Var A, B: String;
    i: Byte;
    k: Word;
    R: String[3];
Begin
    ReadLn(A);
    ReadLn(B);
    R := 'Yes';
    i := 1;
    While (i <= Length(B)) And (R = 'Yes') Do
    Begin
        k := 1;
        While k <= Length(A) Do
            If B[i] = A[k] Then
            Begin
                A[k] := ' ';
                k := Length(A) + 2;
            End
            Else k := k + 1;
        If k = Length(A) + 1 Then R := 'No';
        i := i + 1;
    End;
    WriteLn(R);
    ReadLn
End.
```

Задачі з Алготестера

Для роботи з рядковими величинами пропонуємо такі задачі:

«Коля, Вася і Теніс» (0031),

«Допоможе чи заб'є?» (0081),

«Назва для покемона» (0132),

«Цікаве листування» (0151),

«Петрик і змійка» (0192),

«Дзідзьо і його пісня» (0541),

«Єрмолай та клавіатура» (1508).

Висновок

Є багато цікавих задач, розв’язання яких вимагає вміння працювати з рядковими даними. Досить часто перетворення цілих чисел у дані рядових типів дуже спрощує алгоритм розв’язування завдяки великій кількості функцій і процедур для рядкових величин мовою Паскаль. Готуючись до участі в олімпіадах, обов’язково треба навчитися працювати з величинами рядкових типів.

Рекомендована література

1. Караванова Т.П. Основи алгоритмізації та програмування: 750 задач з рек. та прикл.: Посіб. – К.: Форум, 2002. – 287 с

2. Мансуров К. Т. Основы программирования в среде Lazarus, 2010. – 772 с

3. Алексеев Е. Р., Чеснокова О. В., Кучер Т. В. Free Pascal и Lazarus: Учебник по программированию – М. : ALT Linux ; Издательский дом ДМК-пресс, 2010. – 440 с.

Зміст