КНВК №129
«ГЛАС»

# Умовні оператори

Кафедра інформаційних технологій

# Поняття умови
# Команда розгалуження
# Команда вибору

## Вчимося розв’язувати задачі

# Поняття умови

Дуже часто в житті складаються ситуації, коли людина стоїть перед вибором. Піти в кіно чи підготуватися до уроку, допомогти матусі у домашній справах чи почитати книжку, взяти з собою парасольку чи залишити її вдома тощо.

Врешті решт найвідоміша дилема «Бути чи не бути?» також ставить людину в ситуацію, коли потрібно зробити вибір. Усі ці проблеми можна вирішити тільки за певних умов. Якщо на уроці буде контрольна робота, скоріш за все переважить підготовка до уроку, якщо книжка не є програмним матеріалом, потрібно допомогти матусі, а якщо на вулиці йде дощ, парасолька є необхідною при виході на вулицю.

# Поняття умови

З точки зору комп'ютера, який завжди повинен розв'язувати задачу досконало, вибір відбувається однозначно, незалежно від часу, стану системи або поганого самопочуття користувача. Забезпечується це перш за все поняттям умови.

З точки зору програмування умовою називається речення, на яке можна дати відповідь «так» або «ні». Більш наукове визначення ***умова – це вираз логічного типу***, тобто це вираз, що може приймати значення «істина» або «хибність». З цієї точки зору речення «Яка зараз на вулиці погода?» не є умовою, а речення «На вулиці йде дощ?» – є.

# Поняття умови

Найпростішим прикладом умови є будь-який вираз, що містить порівняння. Наприклад, **x<5, sin(x)+cos(x)>2*sin(x), 5=3.**

Останній вираз, правда, викликає непорозуміння і бажання сказати, що він є неправильним, але давайте трохи поміркуємо. Цей вираз можна трактувати, як питання «число 5 дорівнює числу 3?». Очевидно, що на таке питання не вагаючись усі дадуть відповідь «ні». Повернемося тепер до поняття умови (умова – це речення, на яке…). Виходячи з цього означення, запропонований вираз є умовою, правда її значення завжди буде хибним. Комп'ютер може визначити «істинний» даний вираз чи ні, а тому запропонований запис буде сприйматися, як умова.

# Поняття умови

- Умовою, виходячи з означення, можна також назвати будь-яку змінну логічного типу (**boolean**). Тобто, якщо описати змінну наступним чином:

```
var f:boolean;
```

то сама змінна f також буде умовою.

Усі наведені приклади є умовами ***простими***. Та в багатьох задачах обійтися простими умовами неможливо. Наприклад, подвійна нерівність $0 < x \leq 10$ (число $x$ належить діапазону $(0,10]$), що застосовується у курсі математики, не може бути записана одним виразом, оскільки два знака нерівності використовувати в одному виразі заборонено. У такому випадку користуються так званими ***складеними*** умовами. Наприклад, вище наведену подвійну нерівність можна записати таким чином: `(x>0)and(x<=10)`.

# Поняття умови

Пояснимо, як формуються складені умови. Для вказаного випадку наші міркування були наступними: щоб число належало вказаному діапазону, воно ***одночасно*** повинно бути більшим **0** ***та*** не меншим за **10**. Тобто навіть у словесному описі прозвучало слово «ТА» (**and** – англійською мовою).

Одночасне справдження умов потрібне не завжди. Наприклад, щоб точка належала одній з осей координатної площини, достатньо, щоб ***хоча б одна*** з її координат дорівнювала **0**. Словесно ми проговорюємо це як: координата **x** дорівнює **0 АБО** (**or** – англійською мовою) координата **y** дорівнює 0. Запис мовою програмування Паскаль виглядає наступним чином:**(x=0) or (y=0).**

При цьому, якщо обидві координати дорівнюють **0**, точка все одно належить осям, причому обом.

# Поняття умови

Достатньо рідко, але все ж таки трапляються випадки, коли потрібно справдження ТІЛЬКИ однієї умови. Наприклад, якби у попередньому випадку постала задача виокремити точку перетину координат і визначити, чи належить точка осям та не належить центру координат, ми б словесно сказали, що потрібно, щоб при нульовому значенні однієї координати інша була ненульовою. Це можна записати таким чином:**((x=0)and(y<>0))or((y=0)and(x<>0)).**

Записані вирази містять у собі крім констант, змінних, дужок та знаків порівнянь, ще так звані *логічні операції* **and** та **or**.

Існує ще одна логічна операція «***виключне або***» (**xor**), яка надає більш «вишуканий» спосіб записати останню наведену умову. Ця операція дає істинний результат тільки тоді, коли одна з наведених умов істинна, а друга хибна, тобто вираз **(x=0)xor(y=0)** буде істинним тоді і тільки тоді, коли одна координата нульова, а друга ненульова.

# Поняття умови

Для трьох вказаних логічних операцій існують так звані таблиці істинності, які пояснюють їх виконання:

| x | y | x and y | x or y | x xor y |
|---|---|---|---|---|
| false | false | false | false | false |
| false | true | false | true | true |
| true | false | false | true | true |
| true | true | true | true | false |

Крім перелічених бінарних операцій (у кожній з цих операцій по два операнди) існує ще одна операція, що має один операнд – унарна операція not. Вона є протилежністю до вказаної умови або її інверсією, тобто вираз «x – не додатне число» можна записати таким чином: not(x>0).

Таблиця істинності цієї операції наступна:

| x | not x |
|---|---|
| false | true |
| true | false |

# Команда розгалуження

Часто задача при різних вхідних даних може мати різні відповіді. Наприклад, квадратне рівняння може мати один корінь, два кореня або взагалі не мати коренів, точка на координатній площині може знаходитись у одній з чотирьох можливих чвертей, а трикутник може бути рівностороннім, рівнобічним, прямокутним, гострокутним, тупокутним тощо. Для обробки різних ситуацій в програмуванні існують два умовних оператора – **if** та **case**, які називаються операторами ***розгалуження.*** Вони дозволяють реалізувати в алгоритмі перевірку деякої умови й обробку ситуації у відповідності до неї.

Хоча обидва ці оператори за своєю суттю є умовними, однак, історично склалося так, що оператором *розгалуження* (умовним оператором) називають оператор **if**, а оператор **case** називають оператором *вибору* або оператором *варіанта*. У подальшому будемо дотримуватися цієї історичної термінології.

# Команда розгалуження

Вибір одного з двох можливих шляхів обчислення задають за допомогою ***оператору розгалуження*** (***умовного оператору***), що має такий вигляд.

**if <умова> then <оператор1> [else <оператор2>];**

Квадратні дужки у запису означають, що відповідна гілка може бути відсутньою, тобто оператор розгалуження можна записати у такому вигляді:

**if <умова> then <оператор1>;**

Перша форма запису оператора називається *повною* (дивись її представлення мовою блок-схем на рис. 1,а), а друга – *скороченою* (див. рис. 1,б).

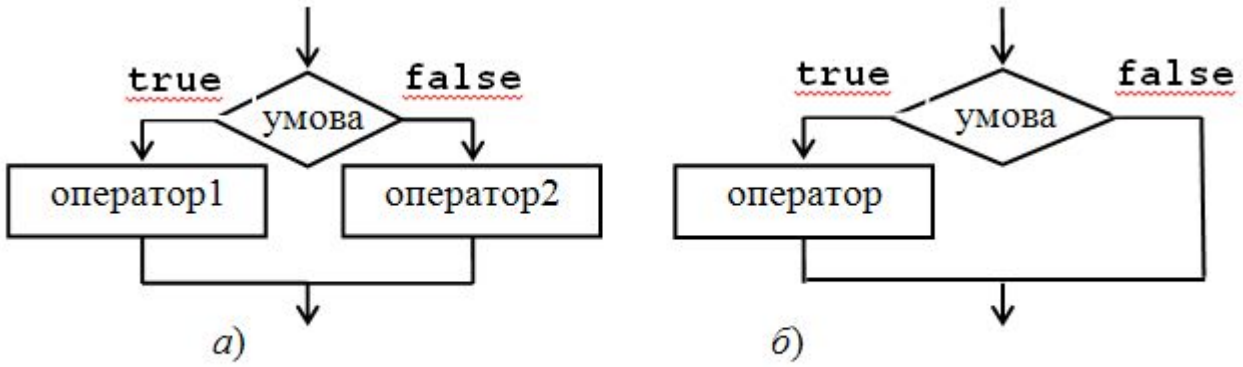



Рис. 1. Блок-схеми двох форм оператора розгалуження

# Команда розгалуження

При виконанні оператора розгалуження спочатку обчислюється значення умови. Якщо отримано значення **true**, то виконується оператор, записаний після слова **then**, і на цьому виконання закінчується. Якщо отримано значення **false**, виконується оператор, записаний після **else**. Обидва оператори довільні і можуть бути як простими, так і складеними.

***Увага!*** Наголосимо ще раз на тому, що оператори розділяються символом **«;»**. Це означає, що **усередині умовного оператора символ «;» зустрічатися не повинен,** інакше усе, що стоїть після нього, буде вважатися іншим оператором. **Перед ключовим словом else символ «;» ніколи не ставиться!**

# Команда розгалуження

Ще одна складність використання оператора **if** виникає при написанні вкладених операторів рогалуження. У випадку, якщо вкладений оператор **if** міститься в межах складеного оператора, великих проблем не виникає, оскільки рятують операторні дужки **begin** ...**end**, а якщо вкладений умовний оператор є єдиним оператором у гілки **else**, виникає неоднозначність: якому **if** відповідає **else**. Наприклад,

**if <умова 1> then if <умова 2>**

**then <оператор1>**

**else <оператор2>;**

У таких випадках слід пам’ятати таке правило:

Ключове слово **else** зв'язується з найближчим, що стоїть перед ним ключовим словом **if**, яке ще не було зв’язане з яким-небудь ключовим словом **else**.

# Команда вибору

Оператор вибору **case** є узагальненням оператора **if.** Він дає можливість виконання одного з декількох операторів в залежності від значення перемикача. Конструкція цього оператору є ідеальним засобом для обробки ситуацій з декількома виходами, коли умова може приймати більше двох значень.

Оператор має такий вигляд:

**case** <вираз-селектор> **of**

<список констант 1>: <оператор 1>;

<список констант 2>: <оператор 2>;

. . .

<список констант N>: <оператор N>;

[**else** <оператор N+1>]

**end**;

# Команда вибору

Оператор вибору **case** є узагальненням оператора **if.** Він дає Як і в умовному операторі квадратними дужками показано, що гілка **else** може бути відсутньою, утворюючи скорочену форму команди вибору.

**case** **<вираз-селектор>** **of**

**<список констант 1>: <оператор 1>;**

. . .

**<список констант N>: <оператор N>;**

**end;**

Оператори, що стоять після двокрапки можуть бути як простими, так і структурованими. У ролі виразу-селектора використовується той, що розташовується між ключовими словами **case** і **of**. Цей вираз може бути тільки порядкового типу, до якого відносяться усі цілі числові типи, крім восьмибайтового, символьні та логічні типи. Всі константи вибору повинні бути *унікальними* і мати тип, сумісний з типом перемикача.

# Команда вибору

Список констант може складатися з однієї, двох і більше констант, які вказуються через кому або, у разі послідовних значень, вказуються два крайніх значення, розділених двома крапками (символом діапазону).

```
case <вираз-селектор> of
  const_one:<оператор>;        // одна константа
  const1, const2: <оператор1>;     // список констант
  min..max: <оператор2>;           // діапазон констант
end;
```

Виконання оператора вибору починається з обчислення виразу-селектора. Якщо результат обчислення дорівнює одній із вказаних констант, виконується відповідний оператор з подальшим передаванням керування за межі оператора вибору. Якщо значення виразу не збігається з жодною з констант, виконується оператор, що стоїть після гілки **else** (у разі її наявності), або управління передається на оператор, який слідує за **end**.

# Вчимося розв’язувати задачі

Авторами була розглянута велика кількість задач із застосуванням команд розгалуження. Деякі з них можна згрупувати за методом розв’язання, а тому вашій увазі пропонується розбиття на категорії, в якому спочатку представлена базова задача з розбором, а потім умови подібних задач. Пропануємо вам розглянути 9 категорій задач:

# І категорія.

До цієї категорії можна віднести задачі, які **у самому формулюванні умови мають алгоритм їх розв'язання**. Розглянемо одну з задач.

***Умова задачі.*** Задано значення дійсних чисел **a, b, c**. Подвоїти ці значення, якщо $a \geq b \geq c$, і замінити їх абсолютним значенням, якщо це не так.

***Розв'язок задачі:***

1. Кожна задача починається з уведення вхідних даних. У нашому випадку потрібно ввести три числа.
2. Використовуючи команду розгалуження перевіряємо умову, що вказана в задачі, та виконуємо задані дії: подвоюємо значення змінних при справдженні умови, та замінюємо їх модулем у протилежному випадку.

Звернуть увагу на те, що оскільки на кожній гілці умовного оператора вимагається виконання трьох команд присвоєння, оператори будуть складеними та вміщуватися в операторні дужки **begin – end**.

# І категорія.

Фрагмент тексту програми має вигляд:

```
var a,b,c:real;
Begin write('Input number '); {Уведення вхідних даних}
readln(a,b,c);
  if (a>=b)and(b>=c) {Команда розгалуження, що відповідає умові}
  then begin  a:=2*a; b:=2*b; c:=2*c;
    end
  else begin
    a:=abs(a); b:=abs(b); c:=abs(c);
    end;
  writeln('a=',a:8:2);  {Виведення результатів}
  writeln('b=',b:8:2);  {у відформатованому вигляді}
  writeln('c=',c:8:2);
  end.
```

# II категорія

- **Обчислити значення функції**.

Ці задачі підрозділяються на дві підгрупи:

- вираз, що підлягає обчисленню, містить операції, які не завжди можна виконати: ділення (ділення на нуль заборонено) або обчислення квадратного кореня (підкореневий вираз не може бути від'ємним);
- для різних значень аргументу обчислення виконуються за різними співвідношеннями.

Розглянемо приклади розв'язання таких задач.

***Задача II.1.*** Обчислити значення функції $y = \frac{\sqrt{2x-10}}{x^2-1}$.

***Розв'язок задачі:*** Очевидно, що обчислення можливі при виконанні двох умов: знаменник не дорівнює **0** та підкореневий вираз невід'ємний. Якщо хоч одна з умов є хибною, виведемо на екран повідомлення про помилку.

# II категорія

Фрагмент тексту програми має вигляд:

```
var x,y:real;
begin   write('Input number  ');   readln(x);
   if (2*x-10>=0)and(sqr(x)-1<>0)
   then begin
      y:=sqrt(2*x-10)/(sqr(x)-1);
      writeln('y=',y:0:2);
   end
   else writeln('Error');
   readln;
end.
```

# II категорія

- *Задача II.2. Дано дійсне число x. Обчислити значення функції*

$$y = \begin{cases} sinx^3, & x > 10; \\ \sqrt{10 - x}, & x \le 10. \end{cases}$$

*Розв'язок задачі: Вибір співвідношення, за яким виконується обчислення функції, залежить від значення аргументу. Якщо x>10, то обчислюємо за першою формулою, в протилежному випадку — за другою.*

*Фрагмент тексту програми має вигляд:*

```
var x,y:real;
begin   write('Input number  ');   readln(x);
   if x>10     then y:=sin(x*x*x)
               else y:=sqrt(10-x);
   writeln('y=',y:0:3);
   readln;
end.
```

# III категорія.

**Знайти максимальне (мінімальне) значення з двох (трьох) чисел**.

***Умова задачі.*** Дано два числа. Знайти серед них число з найбільшим значенням.

***Розв'язок задачі:***

У запропонованій задачі вперше ми стикнулися з ситуацією, коли в умові не оговорені типи вхідних значень. Такі задачі зустрічаються досить часто, оскільки програмування, як прикладна дисципліна, вимагає від програміста вміння аналізувати можливі варіанти вхідних даних та обробляти усі ситуації, навіть такі, що здаються неможливими з точки зору «здорового глузду». Адже користувачем програми іноді може бути людина недосвідчена, а іноді і така, що навмисно буде намагатися привести до аварійного завершення програми.

Запропонована задача, очевидно, повинна вміти обробляти будь-які числа (цілі або дійсні), а тому всі змінні для програми має сенс вибирати дійсними, щоб надати можливість працювати як з цілими (вони є підмножиною дійсних чисел) так і з дробовими числами.

# III категорія.

Крім того, зауважимо, що має сенс увести ще одну змінну, яка буде зберігати знайдене максимальне значення, оскільки в більшості випадків сам пошук максимуму буду підзадачею деякої іншої задачі, що використовуватиме знайдене найбільше значення для подальших обрахунків. Зверніть також увагу, що при рівних числах з точки зору програмування вони обидва є максимальними, а тому така ситуація окремо не обробляється і максимуму надається будь-яке з двох значень.

Фрагмент тексту програми має вигляд:

```
var x,y,max:real;
begin   write('Input numbers  ');   readln(x,y);
  if x>y {Умову можна переформулювати, як x>=y}
        then max:=x   else max:=y;
  writeln('max=',max:0:2);
  end.
```

# III категорія.

При пошуку максимуму з трьох чисел можна отримати розв'язок трьома методами. Пропонуємо подумати, який із запропонованих нижче розв'язків буде оптимальнішим за часом (менша кількість зайвих перевірок).

*1 метод*. Трьох чисел максимальне може бути тільки одне з них, а тому можна отримати три ситуації, які обробляються скороченими оператора розгалуження:

```
if (x>=y)and(x>=z) then max:=x;
if (y>=x)and(y>=z) then max:=y;
if (z>=x)and(z>=y) then max:=z;
```

*2 метод*. При пошуку максимального значення з трьох чисел можна спочатку знайти максимум з двох чисел, а потім максимум зі знайденого максимуму та третього числа. Тобто

```
if x>y   then max:=x   else max:=y;
if max<z   then max:=z;
```

# III категорія.

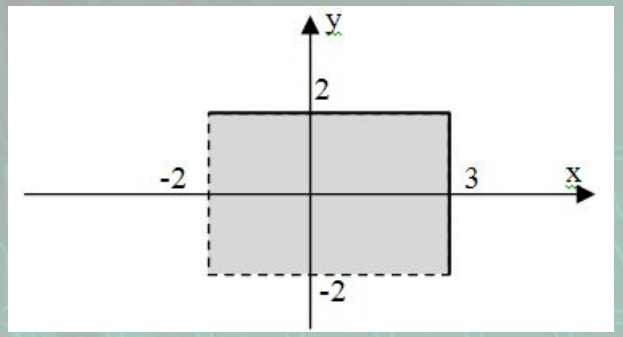
*3 метод*. Підхід пошуку максимуму не змінюється відносно попереднього випадку, змінюється тільки форма запису:

```
if x>y
 then {На цій гілці більшим з двох (x або y) є x}
      if x>z
      then max:=x
      else max:=z
 else {На цій гілці більшим з двох (x або y) є y}
      if y>z
      then max:=y
      else max:=z;
```

# IV категорія

**Задачі про координатну площину**. Більшість цих задач пов'язана з визначенням належності точки визначеної області координатної площини.

***Умова задачі.*** Не будемо розглядати очевидні задачі типу, якій чверті координатної площини належить задана точка, а розглянемо задачу з більш складною областю, що відмічена штриховкою. Наприклад таку:




***Розв'язок задачі:*** Задачі такого типу розв'язуються, виходячи з наступних міркувань: щоб точка лежала на прямій, потрібно справдження рівності, в якій координати точки підставляються у рівняння заданої прямої. Відповідно, якщо точка не лежить на прямій, рівність перетворюється на нерівність, причому нижче прямої у нерівність типу

# IV категорія

• $y < f(x)$, а якщо вище, то $-y > f(x)$, де $f(x)$ – функція, що описує рівняння прямої. Рівняння прямої можна отримати кількома методами, але найпростіший – це використання рівняння прямої, що проходить через дві точки з координатами $(x_1, y_1)$ та $(x_2, y_2)$ : $\frac{x-x_1}{x_1-x_2} = \frac{y-y_1}{y_1-y_2}$

Запропонована задача дозволяє рівняння прямої взагалі отримати без вказаного співвідношення, оскільки прямі є горизонтальними або вертикальними. Для верхньої горизонталі це рівняння буде $y=2$, для нижньої – $y=-2$, для лівої вертикалі $x=-2$, а для правої $x=3$.

Точка буде лежати всередині заштрихованої області, якщо справджуються усі нерівності для чотирьох прямих. Зверніть увагу, що намальовані прямі бувають штриховими, а бувають суцільними. В першому випадку вважається, що точка, яка лежить на прямій, не належить заштрихованій області, а в другому – належить. Відповідно, в першому випадку нерівність буде строга, а в другому – нестрога.

# IV категорія

- *Оскільки результатом розв'язка задачі буде відповідь «належить» або «не належить» точка вказаній області, будемо виводити на екран повідомлення «Yes» або «No» відповідно.*

*Фрагмент тексту програми має вигляд:*

```
var x,y:real;
begin   write('Input point  ');   readln(x,y);
   if (x>-2)and(x<=3)and(y>-2)and(y<=2)   then writeln('Yes')
   else writeln('No');
nd.
```

*У якості ліній, які обмежують задану область можуть бути також графіки функції. Підхід до розв'язання такої задачі не змінюється, тільки у якості функції будуть використовуватися відповідні функціональні залежності. Нагадаємо, що круг має наступну функціональну залежність* $(x-a)^2+(y-b)^2=R^2$, *де* $(a,b)$ *– координати центра круга, а* $R$ *– його радіус.*

# V категорія.

**Задачі на перевірку властивостей цілих чисел**. Майже всі ці задачі передбачають розбиття числа на окремі цифри з наступним їх аналізом.

***Умова задачі.*** Дано чотирицифрове натуральне число **N**. Перевірити, чи є це число:

- паліндромом (число, яке читається з обох боків однаково, наприклад, 1331);
- щасливим (сума перших двох цифр числа дорівнює сумі других двох, наприклад, 1203);
- таким, що всі його цифри парні.

***Розв'язок задачі:***

Ці задачі, по-перше, також вимагають перевірки коректності вхідних даних, оскільки розбиття числа на цифри залежить від їх кількості. По-друге, щоб розбити число на цифри, потрібно застосувати операції цілочисельного ділення та знаходження залишку від цілочисельного ділення числа на 10.

# V категорія.

Так, наприклад, операція **N mod 10** дозволяє отримати останню цифру числа (цифру одиниць), оскільки залишок від ділення числа на 10 дорівнює саме цій цифрі, а операція **N div 10** дозволяє «відкинути» цифру від числа. Не пояснюючи докладно усі команди, наведемо ті, які дозволяють отримати з чотирицифрового числа його окремі цифри (цифри нумеруються справа наліво):

**c1:=N mod 10;**

**c2:=(N div 10) mod 10;**

**c3:=(N div 100) mod 10;**

**c4:=N div 1000;**

Далі розв’язок стає очевидним, оскільки потрібно або порівняти відповідні цифри, або знайти суму деяких цифр, або перевірити властивості самих цифр.

# V категорія.

Фрагмент тексту програми має вигляд:

```
var c1,c2,c3,c4:byte; N:word;
Begin    write('Input numbers  ');   readln(N);
  if (N<1000)or(N>9999) {Перевірка коректності вхідних даних}   then
              writeln('Input error')   else begin
                 c1:=N mod 10;        c2:=(N div 10) mod 10;
               c3:=(N div 100) mod 10;         c4:=N div 1000;
           if (c1=c4)and(c2=c3   {Крайні та середні цифри рівні між собою}
            then writeln('Palindrom')     else writeln('Not palindrom');
      if c1+c2=c3+c4     {Сума перших двох цифр дорівнює сум двох останніх}
      then writeln('Happy)    else writeln('Not happy);
     if (c1 mod 2 = 0)and(c2 mod 2 = 0)and(c3 mod 2 = 0)and(c4 mod 2 = 0)
      {кожна цифра числа є парною}
      then writeln('Numbers parity')   else writeln('Numbers not parity');
  end;
end.
```

# VI категорія

**Геометричні задачі.** Ці задачі також можна умовно поділити на задачі кількох підгруп:

- задачі на можливість існування певної фігури – трикутника, квадрата, ромба, прямокутника тощо;
- задачі на різновиди трикутників – прямокутний, гострокутний, тупокутний або рівносторонній, рівнобічний, різносторонній;
- належність точки прямій.

***Задача V.1.*** Дано три числа **a**, **b** та **c**. З'ясувати, чи можна побудувати трикутник, довжини сторін якого дорівнюють цим числам.

***Розв'язок задачі:***

У задачах даної категорії вперше з'являється поняття некоректних вхідних даних. Так, наприклад, довжини сторін не можуть бути від'ємними або нульовими. А тому після уведення вхідних даних обов'язково потрібно перевірити їх коректність і, якщо дані некоректні, вивести про це повідомлення.

# VI категорія

Спроба використати беззнакові цілі типи для збереження вхідних даних не є слушною, оскільки довжини довільних трикутників можуть мати дробові значення.

Тепер розглянемо математичну модель задачі. Очевидно, що навіть маючи три додатних числа, ми не завжди зможемо побудувати трикутник з такими сторонами. З курсу математики учням відоме таке правило: трикутник можна побудувати тоді і тільки тоді, коли сума довжин двох менших сторін буде більшою за довжину третьої сторони. Часто стереотип мислення помилково підказує, що меншими повинні бути сторони **a** та **b**, а **c** – це найбільша сторона. І тому на прикладі цієї задачі потрібно привчати учнів розглядати всі можливі випадки і перевіряти всі умови. Для даної задачі можливі три випадки, коли найбільшою буде одна з трьох сторін і тому повний розв'язок буде містить складену умову з трьох простих умов, поєднаних логічною операцією **and** (одночасне справдження трьох умов). Відповідь у цій задачі, як і раніше, буде у вигляді «**YES**» або «**NO**» в залежності від того можна чи не можна побудувати трикутник.

# VI категорія

Фрагмент тексту програми має вигляд:

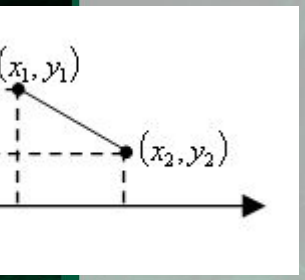
```
var a,b,c:real;
begin   write('Input numbers  ');   readln(a,b,c);
      if  (a<=0)or(b<=0)or(c<=0)      {Перевірка коректності вхідних даних}   then writeln('Input error')
      else if (a+b>c)and(a+c>b)and(c+b>a)
            then writeln('YES')    else writeln('NO');
   readln;
end.
```

Зауважимо, що теоретично відповідь можна виводити будь-якими літерами (великими чи малими), оскільки людина сприймає їх зміст однаково. Але, готуючись до машинної перевірки розв'язків, можна привчати учнів до строгості навіть у цьому, чітко оговорюючи, якими саме літерами потрібно виводити результат.

# VI категорія

- Очевидно, що фігуру можна задавати не тільки довжинами сторін. Можна ще задати її координатами вершин. У цьому випадку, використовуючи відомі формули обчислення довжини відрізка заданого координатами його кінців, можна знайти довжини сторін та звести задачу до попередньої. Згідно малюнка довжина відрізка обчислюється за формулою $a = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

Некоректних даних у цьому випадку бути не може, але точки можуть лежати на одній прямій, утворюючи вироджену фігуру. Перевірку, чи лежать точки на одній прямій ми розглянемо трохи пізніше.




Трикутник також може бути задано величинами його кутів.

Некоректність вхідних даних у цьому випадку перевіряється так само, як і довжин сторін, а можливість існування трикутника забезпечується відомим математичним співвідношенням, що сума кутів трикутника повинна дорівнювати $180^0$. Для чотирикутника перевірка відбувається так само, але математична модель трохи інша:

- для чотирикутника, що заданий довжинами сторін, сума довжин трьох сторін більше четвертої;
- для чотирикутника, що заданий величинами кутів, сума кутів чотирикутника дорівнює $360^0$.

# VI категорія

***Задача V.2.*** Дано три числа **a**, **b** та **c**. Визначити чи є трикутник, довжини сторін якого дорівнюють заданим числам, прямокутним.

***Розв'язок задачі:***

Задачі даного типу є продовженням попереднього, оскільки її повний розв'язок передбачає спочатку перевірку можливості існування даного трикутника. Зверніть увагу, що теоретично задані числа можуть бути і кутами трикутника. І крім того, можна ще задавати трикутник координатами точок на площині. У перших двох випадках (сторони і кути) потрібно обов'язково перевіряти коректність вхідних даних.

Якщо трикутник існує, то визначення, чи є він прямокутним, виконується з використанням теореми Піфагора. Зауважимо при цьому, що шкільний стереотип, який наголошує, що **a** та **b** – катети, а **c** – гіпотенуза, є необов'язковим у програмуванні. А тому потрібно перевірити три можливих випадки справдження цього рівняння.

# VI категорія

Фрагмент тексту програми має вигляд:

```
var a,b,c:real;
Begin    write('Input numbers  ');   readln(a,b,c);
  if (a<=0)or(b<=0)or(c<=0)   {Перевірка коректності вхідних даних}
  then writeln('Input error')
  else if (a+b>c)and(a+c>b)and(c+b>a)  {Перевірка можливості
побудови}
     then if (a*a+b*b=c*c)or(a*a+c*c=b*b)or(c*c+b*b=a*a)
          then writeln('Right triangel')
          else writeln('Not right triangel')
      else writeln('NO');
  end.
```

# VI категорія

Фрагмент тексту програми має вигляд:

```
var a,b,c:real;
Begin    write('Input numbers  ');   readln(a,b,c);
  if (a<=0)or(b<=0)or(c<=0)   {Перевірка коректності вхідних даних}
  then writeln('Input error')
  else if (a+b>c)and(a+c>b)and(c+b>a)  {Перевірка можливості
побудови}
     then if (a*a+b*b=c*c)or(a*a+c*c=b*b)or(c*c+b*b=a*a)
          then writeln('Right triangel')
          else writeln('Not right triangel')
       else writeln('NO');
  readln;
end.
```

# VI категорія

***Задача V.3.*** Дано координати трьох точок на координатній площині. Визначити, чи лежать ці точки на одній прямій.

***Розв'язок задачі:*** Для розв'язання даного класу задач знову повернемось до рівняння прямої, що проходить через дві точки:

Очевидно, що якщо всі задані точки лежать на одній прямій, підстановка координат третьої точки в рівняння прямої повинно перетворити рівняння у рівність. Рекомендуємо при цьому позбавитись у рівнянні операції ділення нескладним алгебраїчним перетворенням:

Фрагмент тексту програми має вигляд:

```
var x1,y1,x2,y1,x3,y3:real;
Begin   write('Input points 1  ');   readln(x1,y1);
  write('Input points 2  ');   readln(x2,y2);
  write('Input points 3  ');   readln(x3,y3);
  if (x3-x1)*(y1-y2)=(y3-y1)*(x1-x2)
  then writeln('Straight line')    else writeln('Not straight line');
  end.
```

# VII категорія.

- **Задачі на розв'язання рівнянь або нерівностей**. Для цих задач спочатку дуже важливою є побудова математичної моделі. Так, для лінійного рівняння $ax = b$ потрібно розглянути наступні випадки:

- $a \neq 0$ у цьому випадку рівняння має єдиний корінь ;
- $a = 0$ у цьому випадку можливі дві ситуації: якщо друге число $b$ дорівнює нулю, рівняння має безліч розв'язків, у протилежному випадку – розв'язків взагалі немає.

А тепер розглянемо іншу часто використовувану задачу.

***Умова задачі***. Дано три числа *a, b, c*. Знайти корені квадратного рівняння $ax^2+bx+c=0$

***Розв'язок задачі:*** Ця задача має багато підходів до розв'язання, які залежать від вхідних даних.

# VII категорія.

- *По-перше, якщо перший коефіцієнт дорівнює нулю, можливі дві реакції:*
- *рівняння не квадратне, вхідні дані некоректні;*
- *рівняння вироджене і тоді в залежності від інших коефіцієнтів можна отримати відповідь «безліч розв'язків» (всі коефіцієнти нульові), «немає розв'язків» (коефіцієнти при змінній нульові, а вільний член — ні) або один розв'язок, що дорівнює* $x = -\frac{c}{b}$.

*По-друге, якщо перший коефіцієнт не нульовий, рівняння не вироджене і розв'язується за алгоритмом, відомим з курсу шкільної математики:*

- *обчислюється дискримінант за формулою* $D = b^2 - 4ac$;
- *якщо дискримінант від'ємний — коренів немає; якщо нульовий — корінь один і обчислюється за формулою* $x = \frac{-b}{2a}$ *; якщо дискримінант додатній — коренів два і обчислюються вони за формулами*

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

# VII категорія.

Запишемо фрагмент тексту програми з урахуванням, що якщо коефіцієнт **a** дорівнює 0, вхідні дані вважаються некоректними:

```
var a,b,c,x1,x2,d:real;
Begin    write('Input numbers  ');   readln(a,b,c);
  if (a=0)     {Перевірка коректності вхідних даних}
  then writeln('Input error')
  else begin  d:=b*b-4*a*c;
       if (d<0)  then writeln('Solution of equation no exist')
         else if d=0  then begin x1:=-b/(2*a);   writeln('x=',x:0:3);  end
          else begin  x1:=(-b+sqrt(d))/(2*a);  x2:=(-b-sqrt(d))/(2*a);
               writeln('x1=',x1:0:3);   writeln('x2=',x2:0:3);
          end;   end;
  end.
```

# VIII категорія

**Задачі з шаховою дошкою.**

***Умова задачі.*** Дано два числа **row** та **col**, що є номером рядка та стовпчика на шаховій дошці. Визначити, якого кольору клітинка з цими координатами.

***Розв'язок задачі:*** Спочатку визначимося, як задаються координати клітинки шахової дошки. На реальній шаховій дошці рядки нумеруються числами від 1 до 8 , а стовпчики задаються буквами від **a** до **h**. Оскільки в програмуванні змішувати різні типи даних досить складно, домовимося, що і рядки, і стовпчики задаються цілими числами від 1 до 8. Таким чином перевірка коректності вхідних даних тут обов'язкова.

Тепер спробуємо проаналізувати колір різних клітинок дошки. Оскільки нижня ліва клітинка з координатами (1,1) біла, то далі білі клітинки першого ряду будуть мати непарні номери стовпчиків. Відповідно клітинки першого ряду з парними номерами стовпчиків будуть чорними. У другому рядочку ситуація протилежна: клітинки з непарними номерами стовпчиків мають чорний колір, а клітинки з парними номерами – білий. Далі картина повторюється.

# VIII категорія

Підіб'ємо підсумок: рядочки з непарними номерами мають білими клітинки з непарними номерами стовпчиків, а рядочки з парними номерами навпаки. Тобто можна сказати, що у білих клітинок парність номерів рядочків і стовпчиків співпадає.

З усього вище сказаного перевірку клітинки на білий колір можна оформити двома способами:

- перевірка парності номерів рядків та стовпчиків окремо;

  **(row mod 2 = 0)and(col mod 2 = 0) or (row mod 2 = 1)and(col mod 2 = 1)**

- співпадання парності номерів рядків та стовпчиків: **(row mod 2 = col mod 2)**

Фрагмент програми має такий вигляд:

```
var row,col:byte;
Begin   write('Input numbers  ');   readln(row,col);
if (row<1)or(row>8)or(col<1)or(col>8) {Перевірка коректності вхідних даних}
   then writeln('Input error') else
   if row mod 2 = col mod 2   then writeln('White')   else writeln('Black');
     end.
```

# IX категорія.

**Задачі, які потребують для свого розв'язання використання команди вибору.**

***Умова задачі.*** Дано натуральне число **N** , яке визначає вік людини (в роках). Додати до цього числа найменування «рік», «роки» або «років», наприклад 1 рік, 42 роки, 26 років.

***Розв'язок задачі:***

Після введення числа з клавіатури обов'язкова, як і раніше, перевірка коректності вхідних даних, інакше відповідь може бути отримана неправильна (подумайте чому).

Далі очевидно, що слово, яке дописується до числівника, залежить від останньої цифри числа. Якщо ця цифра «1» – дописується слово «рік», якщо цифри «2» або «3» – дописується слово «роки», в усіх інших випадках – дописується слово «років». Винятком є числа другого десятка, оскільки до них завжди дописується слово «років» незалежно від останньої цифри. Ці числа потрібно перевірити окремо командою умовного переходу.

# IX категорія.

Відповідь має три варіанти, а тому для наочності має сенс використовувати команду вибору. Зверніть увагу, що оскільки кириличний текст виводиться на консоль некоректно, для виведення застосовується транслітерація.

Фрагмент програми має такий вигляд:

```
var Year:byte;
begin   write('Input year  ');   readln(year);
 if year>100 {Перевірка коректності уведення}  then
writeln('Input.error.')
      else if (year>=10)and(year<=20) then writeln(Year,'  rokiv.')
          else case Year mod 10 of
               1: writeln(Year,'  rik.');
               2..4: writeln(Year,'  roky.');
               else writeln(Year,'  rokiv.')
             end;
end.
```

КГТЛ №129

# Перевір себе!

# Матеріал для презентації

За матеріалом статті «Що буде, якщо…?» Автори М. Єгорова, І.Скляр (ж.Інформатика №10, 11 2013 р.. Видавництво «Основа» м.Харків)