Поняття про булеву логіку. Логічні (булеві) операції та операції відношення (порівняння) Формування умов

Одними з найважливіших у програмуванні є умовні оператори. Вони наявні практично в кожній програмі і здійснюють масу дуже важливих перевірок — виводити чи не виводити, використати той чи інший фрагмент алгоритму, повідомляти про помилку чи не повідомляти. Цей розділ присвячений питанням алгебри логіки й логічним операторам, побудованим на її основі.

Джордж Буль (George Boole), засновник математичної логіки, народився 2 листопада 1815 р. в місті Лінкольні (Велика Британія) у родині небагатого ремісника Джона Буля. Перші уроки математики майбутньому генієві дав його батько. Однак у юні роки Буль не виявив талантів у точних науках — його першим захопленням стала класична література. У 12 років Джон знав латину, потім оволодів грецькою, французькою, німецькою й італійською мовами. Лише в сімнадцять років Буль захопився вищою математикою. У 20 років Дж. Буль відкрив власну школу в Лінкольні. Усе своє життя він працював у школах та коледжах.

Джордж Буль — автор 50 статей на математичні теми й кількох монографій, що стали класичними. Інтереси Буля в математиці були різнобічними, але всесвітню славу йому принесла математична логіка. Можна вважати, що Буль був першим математиком, який звернувся до логічної проблематики.

У своєму дослідженні (1844), опублікованому у «Філософських працях Королівського товариства», Буль уперше пише про «обчислення висловів». 1847 р. він опублікував памфлет «Математичний аналіз логіки», у якому висловив ідею, що логіка ближча до математики, аніж до філософії. Цю роботу дуже високо поцінував шотландський математик Августус де Морган. Завдяки їй Буль у 1849 р. обійняв посаду професора математики Квінз-коледжу в графстві Корк, незважаючи на те що він навіть не мав університетської освіти.

Однак памфлет не міг вважатися серйозною науковою працею, і Буль повною мірою висловлює свої погляди на цю проблему в трактаті «Дослідження законів мислення, на яких ґрунтуються математичні теорії логіки та ймовірностей» (1854).

Одиницею Буль позначав множину всіх мислимих об'єктів, літерними символами — вибірки з неї, пов'язані зі звичайними іменниками й означеннями (так, якщо x= «рогаті», а *y* = «вівці», послідовна вибірка *x* і *y* з одиниці дасть клас рогатих овець). Буль показав: символи такого роду підпорядковуються тим самим законам, що й алгебраїчні, тож будь-які об'єкти можна додавати, віднімати і множити (так, різниця *1-x* являє собою операцію вибірки з універсуму всіх безрогих об'єктів, а добуток (1-x)(1- y) — вибірку об'єктів, що не є ані рогатими, ані вівцями).

Роботи Дж. Буля 1847 і 1854 рр. започаткували алгебру логіки — згодом булеву алгебру. Учений, скориставшись двійковою системою числення, придумав свою систему позначень і правил, за допомогою яких можна закодувати будь-які вислови, а потім маніпулювати ними як звичайними числами, передбачаючи тільки два варіанти відповіді — «істина» або «хибність» (у Delphi — True і False), «одиниця» або «нуль». Булева алгебра має три основні операції — І, АБО, НЕ, які дозволяють здійснювати множення, додавання й заперечення логічних умов.

Створюючи мову Pascal, Ніклаус Вірт увів для позначення логічного типу даних зарезервоване слово Boolean на пошану творця математичної логіки. Цікаво, що в першому стандарті мови це слово належало писати з великої літери.

Отже, математична логіка (булева алгебра) оперує логічними змінними, які можуть набувати значень двох видів — true (істина) і false (хибність). У числовому кодуванні «хибність» задається цілочисловим значенням 0, а «істина» — числовим значенням 1 або більше. Формально ми можемо «спитати» в програми: «Чи ІСТИНА більша за ХИБНІСТЬ?» — і одержати ствердну відповідь.

Значення логічним змінним задаються шляхом присвоювання констант або результатів порівняння. Розглянемо приклад:

```
var
  a: integer; x,y: Boolean;
begin x:=true; a:=7;
  y:=a > 2;
```

Обидві логічні змінні (*x* і *y)* після виконання цього фрагмента програми набудуть значення true («істина»).

В операціях відношення застосовуються звичні в математиці знаки порівняння: > («більше»), < («менше»), = («дорівнює»), >= («більше або дорівнює»), <= («менше або дорівнює»), <> («не дорівнює»). Три останні знаки відрізняються від звичайного математичного запису. Це зумовлене історичними причинами: старі принтери й алфавітно- цифрові монітори не могли відтворювати знаки ^ , * і <, тому для них було придумано заміну з комбінацій символів, що виводяться в один рядок.

Як правило, умови порівняння в розв'язуваних задачах набагато складніші, ніж проста перевірка на рівність або нерівність. Навіть прості задачі часто вимагають одночасного виконання відразу кількох умов. Розглянемо класичний приклад.

Відомо, що Земля обертається навколо Сонця не за 365 діб, а за 365,242199 доби. Якщо календарний рік становить 365 днів, то за 4 роки «набігає» зайва доба (точніше, «майже» доба, бо 0,242199 усе-таки менше за чверть доби). Подолати цю складність досить просто. Творці сонячних календарів установили, що кожен четвертий рік є високосним, тобто складається з 366 днів. Як уже йшлося в розділі про алгоритмічні структури, у Delphi є оператор, що може виконувати ту чи іншу дію залежно від логічної умови. Умова «високосності» року в цьому операторі записується так:

```
year:=2010;
if year mod 4=0 then
   Labell.Caption:='Високосний рік'
else
   Labell.Caption:='Невисокосний рік';
```

Однак такий підхід призводить до поступового накопичення помилок. Тому раз у 400 років один високосний рік прийнято вважати невисокосним. Визначається він так: це рік, що закінчується на два нулі «00», перші дві цифри якого діляться на 4. Тепер нам в умовному операторі слід написати такий логічний вираз: «номер року ділиться на 4 без остачі, і при цьому, якщо він ділиться без остачі на 100 (тобто закінчується на «00»), число його сотень не ділиться на 4». Умова занадто громіздка, трохи простіший вигляд матиме обернений варіант — умова «невисокосності» року. Записується вона так:

```
year:=2010;
if (year mod 4<>0) or
    (year mod 100=0)and((year div 100) mod 4<>0) then
     Labell.Caption:='Невисокосний рік'
else
    Labell.Caption:='Високосний рік';
```

Застосовані у виразах логічні операції **or (АБО)** та **and (І)** ввів у математичну логіку ще Буль.

У всіх мовах програмування реалізуються щонайменше три логічні операції:

- логічне заперечення — **not**, заміна «істини» на «хибність» і навпаки (наприклад, a:=10; x:=not(a > 2); після виконання цих операторів *x* дорівнюватиме false);
- логічне множення — **and**, вимагає одночасного виконання обох умов (наприклад, a:=10; x:=(a>2)and(a<12); дає в результаті значення x, що дорівнює true; у випадку, якщо a:=14, x дорівнюватиме false);
- логічне додавання — **or,** вимагає виконання хоч однієї з двох умов (наприклад, a := 5 ; x := (a < 7)or(a > 10); дає в результаті значення x, що дорівнює **true,** хоч умова a > 10 тут не виконується);
- «виключне АБО» — **xor,** дає в результаті «істину», якщо значення операндів збігаються («істина» і «істина», «хибність» і «хибність»).

У математиці значення логічних операцій прийнято подавати у вигляді таблиць істинності, які виводять результати логічних операцій для всіх можливих значень аргументів (табл. 8.1).

Таблиця 8.1

Таблиці істинності для логічних операцій, наявних у Delphi

Змінна x	False	False	True	True
Змінна y	False	True	False	True
not X	True	True	False	False
not y	True	False	True	False
x **or** y	False	True	True	True
x **and** y	False	False	False	True
x **xor** y	True	False	False	True

□ ЗАВДАННЯ ДЛЯ САМОСТІЙНОГО РОЗВ'ЯЗУВАННЯ

Внести в табл. 8.2 значення True або False згідно зі зразком (як це зроблено в першому рядку).

Таблиця 8.2

Результати логічних операцій

Значення змінної a	1	3	5	7	9
Значення змінної b	10	-1	2	20	7
(a **mod** 2 = 0) **or** (b < 10)	False	True	True	False	True
((a **mod** 2) = (a **div** 2)) **or** (b>0)					
(not (a > 2)) **and** (b < 11)					
(a <= 2) **or** (a > 7) **or** (b > a)					
(a >= 5) **and** (b **mod** 6 > 2)					
(a * b > 10) **and** (a > b)					
(a > 2 * b) **or** (b > a * 2)					

ТЕОРЕТИЧНІ ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

- У чому полягають досягнення математика Дж. Буля?
- Якими змінними оперує математична логіка?
- Перелічіть можливі операції відношення (порівняння).
- Назвіть чотири основні логічні операції.
- Що більше — True чи False?

Формування умов

Умови для виконання тієї чи іншої дії можуть формуватися шляхом порівняння обраних значень, як результат обчислення логічних виразів або шляхом опитування відповідних відеокомпонентів. Найбільш простим і поширеним компонентом для введення умов одно- або двоальтернативного розгалуження є компонент CheckBox розташований на закладці Standard.

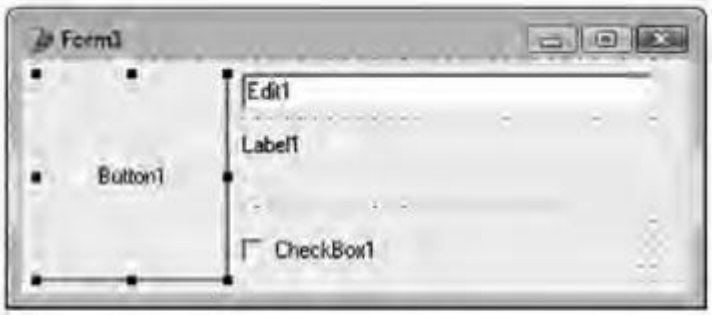

Рис. 8.1. Вигляд форми, одержаний у процесі візуального проектування

На формі цей компонент має вигляд квадратної комірки з підписом; клацанням мишею в цю комірку можна помістити відповідну позначку (за замовчуванням — «галочку»). У процесі виконання програми наявність

або відсутність позначки в елементі CheckBox перевіряється (а в разі потреби встановлюється) властивістю Checked.

Розглянемо простий приклад.

Приклад 5. На формі в Edit вводиться довжина сторони квадрата. Натисканням кнопки в Label виводиться його площа. Якщо в розташованому на формі компоненті CheckBox стоїть позначка, до числового значення в Label приписується напис «кв. м» (рис. 8.1).

У процесі готування завдання настроїмо відеокомпоненти: замінимо рядки Caption на кнопці й компоненті CheckBox на Вигляд форми, одержаний значущі написи. У Label рядок Caption візуального проектування приберемо, у компоненті Edit очистимо властивість Text (рядок, що вводиться,— за замовчуванням там назва компонента).

Подвійним клацанням на кнопці перейдемо в режим введення програмного коду в метод TForml. ButtonlClick, тобто запишемо фрагмент коду, що буде виконуватися в разі натискання кнопки:

```
procedure TForml.ButtonlClick(Sender: TObject);
begin
Labell.Caption:=floattostr(sqr(strtofloat(editl .Text))); if
CheckBoxl.Checked then Labell.Caption:=Labell.Caption+'кв.м';
end;
```

Коли запустити програму на виконання, вона працюватиме так, як показано на рис. 8.2.

Зверніть увагу: для зручності читання написи в усіх компонентів виконані напівжирним шрифтом (властивість Font.Bold має значення

а б

Рис. 8.2. Робота програми з непозначеним (*а*) і позначеним (*б*) CheckBox

True). Програму можна вдосконалити, додавши ще один CheckBox для перемикання звичайного і напівжирного шрифтів (це можна зробити за допомогою окремої кнопки, а можна додати програмний код безпосередньо в кнопку «Обчислення»):

```
procedure TForml.ButtonlC1 ick(Sender:
TObject); begin
  Labell .Caption :=floattostr(sqr(strtofloat (editl .Text))); if
  CheckBoxl.Checked then
    Labell.Caption:=Labell.Caption+' кв. м';
  if CheckBox2.Checked then begin
    Buttonl.Font.Style:=[fsBold];
  Labell.Font.Style:=[fsBold];
  Editl.Font.Style:=[fsBold];
  CheckBoxl.Font.Style:=[fsBold];
  CheckBox2.Font.Style:=[fsBold];
  end else begin
    Buttonl.Font.Style:=[];
    Labell.Font.Style:=[];
    Editl.Font.Style:=[];
    CheckBoxl.Font.Style:=[];
    CheckBox2.Font.Sty1e:=[];
    end; end;
```

Примітка. І перша, і друга програми не здійснюють перевірки на відсутність числового значення в компоненті Edit. Якщо, на ваш погляд, таку перевірку треба передбачити, напишіть цей оператор IF самі.

Коли запустити програму на виконання, вона тепер працюватиме, як показано на рис. 8.3.

а б

Рис. 8.3. Виконання програми з позначкою в CheckBox «Виводити напівжирним шрифтом» (а) і без позначки *(б)*.

ТЕОРЕТИЧНІ ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

- Як можуть формуватись умови для виконання якої-небудь дії?

- Для чого призначений компонент CheckBox?
- Назвіть основну властивість компонента CheckBox. Як вона використовується?
- Назвіть властивість компонента CheckBox, що змінюється в разі позначення елемента хрестиком. Яке значення (True чи False) буде записане в цю властивість?
- Яка властивість компонента CheckBox відповідає за виведення тестового запитання поряд із діалоговим елементом?
- Опишіть, які властивості слід настроїти, щоб написи в Label і CheckBox виконувалися більшим напівжирним шрифтом.