Оператори мови C

Блок або складовою оператор

За допомогою фігурних дужок кілька операторів мови, можливо перемежовуються оголошеннями, можна об'єднати в єдину синтаксичну конструкцію, звану блоком або складеним оператором:

```
{
оператор_1
...
оператор_N
}
```

Синтаксично блок сприймається як одиничний оператор і може всюди використовуватися в конструкціях, де синтаксис вимагає одного оператора. Тіло циклу, гілки оператора if, як правило, представляються блоком.

порожній оператор

Порожній оператор - це порожньо, що завершується крапкою з комою. Іноді корисно розглядати відсутність операторів як існуючий порожній оператор. Синтаксично допустимо ставити зайві крапки з комою, вважаючи, що вставляються порожні оператори. Наприклад, синтаксично допустима наступна конструкція:

```
for (int j = 1; j <5; j ++)
{;;;};
```

Ця конструкція може розглядатися як затримка за часом, робота на холостому ході.

Оператори вибору

Як в C ++ та інших мовах програмування, в мові C # для вибору однієї з кількох можливостей використовуються дві конструкції - if і switch. Першу з них зазвичай називають альтернативним вибором, другу - розбором випадків.

оператор if

Почнемо з синтаксису оператора if:

if (вираз_1) оператор_1
else if (вираженіе_2) оператор_2
...
else if (вираженіе_K) оператор_K
else оператор_N

Які особливості синтаксису слід зазначити? Вирази if повинні полягати в круглі дужки і бути булевского типу. Точніше вираження повинні давати значення true або false. Нагадаю, арифметичний тип не має явних або неявних перетворень до булевскому типу. Слідуючи синтаксису мови C ++, then-гілка оператора слід відразу за круглою дужкою без ключового слова then, типового для більшості мов програмування. Кожен з операторів може бути блоком, зокрема if-оператором. Тому можлива і така конструкція:

if (вираження1) if (вираженіе2) if (вираз3) ...

Гілки else if, що дозволяють організувати вибір з багатьох можливостей, можуть бути відсутніми. Може бути опущена і заключна else-гілка. В цьому випадку коротка форма оператора if задає альтернативний вибір - робити або не робити - виконувати чи не виконувати then-оператор.

Семантика оператора if проста і зрозуміла. Вирази if перевіряються в порядку їх написання. Як тільки отримано значення true, перевірка припиняється і виконується оператор (це може бути блок), наступний за виразом, який отримав значення true. Із завершенням цього оператора завершується і оператор if. Гілка else, якщо вона є, відноситься до найближчого відкритого if.

оператор switch

Приватним, але важливим випадком вибору з декількох варіантів є ситуація, при якій вибір варіанта визначається значеннями деякого виразу. Відповідний оператор C #, успадкованих від C ++, але з невеликими змінами в синтаксисі, називається оператором switch. Ось його синтаксис:

switch (вираз)

{
case константное_вираженіе_1: [оператори_1 оператор_перехода_1]
...
case константное_вираженіе_K: [оператори_K оператор_перехода_K]
[Default: оператори_N оператор_перехода_N]
}

Гілка default може бути відсутнім. Зауважте, по синтаксису допустимо, щоб після двокрапки слідувала порожня послідовність операторів, а не послідовність, що закінчується оператором переходу. Вирази зі сталими в case повинні мати той же тип, що і switch-виразу.

Семантика оператора switch трохи заплутана. Спочатку обчислюється значення switch-виразу. Потім воно по черзі в порядку проходження case порівнюється на збіг з константними виразами. Як тільки досягнуто збіг, виконується відповідна послідовність операторів case-гілки. Оскільки останній оператор цієї послідовності є оператором переходу (найчастіше це оператор break), то зазвичай він завершує виконання оператора switch. Використання операторів переходу - це погана ідея. Таким оператором може бути оператор goto, передає управління іншій case-гілки, яка в свою чергу може передати управління ще куди-небудь, отримуючи блюдо «спагетті» замість добре структурованої послідовності операторів. Семантика ускладнюється ще й тим, що case-гілка може бути порожньою послідовністю операторів. Тоді в разі збігу константного виразу цієї гілки зі значенням switch-виразу буде виконуватися перша непорожній послідовність черговий case-гілки. Якщо значення switch-виразу не збігається ні з одним константним виразом, то виконується послідовність операторів гілки default, якщо ж такої гілки немає, то оператор switch еквівалентний порожньому оператору.

Вважаю, що оператор switch це найбільш невдалий оператор мови C # як з точки зору синтаксису, так і семантики. Невдалий синтаксис породжує

заплутану семантику, що є джерелом поганого стилю програмування. Зрозуміти, чому авторів спіткала невдача, можна, виправдати - ні. Справа в тому, що оператор успадкований від C ++, де семантика і синтаксис цього оператора ще гірше. У мові C # синтаксично кожна case-гілка повинна закінчуватися опера тором переходу (забудемо на хвилину про порожній послідовності), інакше виникне помилка періоду компіляції. У мові C ++ це правило не є синтаксично обов'язковим, хоча на практиці застосовується та ж конструкція з кінцевим оператором break. При його відсутності управління «провалюється» в наступну case-гілка. Звичайно, професіонал може з успіхом використовувати цей трюк, але в цілому ні до чого хорошого це не призводить. Борючись з цим, в C # зажадали обов'язкового включення оператора переходу, завершального гілка. Набагато краще було б, якби останнім оператором міг бути тільки оператор break, писати його було б не потрібно, і семантика стала б прозорою - при збігу значень двох виразів виконуються оператори відповідної case-гілки, при завершенні якої завершується і оператор switch.Еще одна невдача в синтаксичної конструкції switch пов'язана з істотним обмеженням, що накладається на case-вирази, які можуть бути тільки константним виразом. Якщо вже змінювати оператор, то набагато краще було б використовувати синтаксис і семантику Visual Basic, де в case-виразах допускається список, кожне з виразів якого може задавати діапазон значеній.Разбор випадків - це часто зустрічається ситуація в самих різних завданнях. Використовуючи оператор switch, пам'ятайте про недоліки його синтаксису, використовуйте його в правильному стилі. Закінчуйте кожну case-гілка оператором break, але не застосовуйте goto.Оператори переходаОператоров переходу, що дозволяють перервати природний порядок виконання операторів блоку, в мові C # несколько.Оператор gotoОператор goto має простий синтаксис і семантику: goto [мітка | case константное_вираженіе | default]; Все оператори мови C # можуть мати мітку - унікальний ідентифікатор, що передує оператору і відокремлений від нього

символом двокрапки. Передача управління позначеного оператору - це класичне використання оператора goto. Два інші способи використання goto - це передача керування в case або default-гілка - використовуються в операторі switch, про що йшла мова вище. «Про шкоду оператора goto» і про те, як можна обійтися без нього, писав ще Едсгар Дейкстра при обгрунтуванні принципів структурного програмірованія.Я вже багато років не застосовую цей оператор і вважаю, що хороший стиль програмування не передбачає використання цього оператора в C # ні в якому зі своїх варіантів - ні в операторі switch, ні для організації безумовних переходов.Оператори break і continueВ структурному програмуванні визнаються корисними «переходи вперед» (але не тому), що дозволяють при виконанні деякої умови вийти з циклу, оператора вибору, з блоку. Для цієї мети можна використовувати оператор goto, але краще використовувати спеціально призначені для цих цілей оператори break і continue.Оператор break може стояти в тілі циклу або завершувати case-гілка в операторі switch. Приклад його використання в операторі switch вже демонструвався. При виконанні оператора break в тілі циклу завершується виконання самого внутрішнього цікла.Оператор continue використовується тільки в тілі циклу. На відміну від оператора break, завершального внутрішній цикл, continue здійснює перехід до наступної ітерації цього цікла.Оператор returnЕще одним оператором, що належать до групи операторів переходу, є оператор return, що дозволяє завершити виконання процедури або функції. Його синтаксис: return [вираз]; Для функцій його присутність і аргумент обов'язкові, оскільки вираз в операторі return задає значення, що повертається функцією.Оператори циклаБез циклів жити не можна в програмах, нет.Оператор forНаследованний від C ++ дуже зручний оператор циклу for узагальнює відому конструкцію циклу типу арифметичної прогресії. Його синтаксис: for (ініціалізатор; умова; список_вираженій) операторОператор, що стоїть після закриває дужки, задає тіло циклу. У більшості випадків тілом циклу є блок. Скільки разів буде

виконуватися тіло циклу, залежить від трьох керуючих елементів, заданих в дужках. Ініціалізатор задають початкове значення однієї або декількох змінних, часто званих лічильниками або просто змінними циклу. У більшості випадків цикл for має один лічильник, але часто корисно мати кілька лічильників, що і буде продемонстровано в наступному прикладі. Умова задає умову закінчення циклу, відповідний вираз при обчисленні має отримувати значення true або false. Список виразів, записаний через кому, показує, як змінюються лічильники циклу на кожному кроці виконання. Якщо умова циклу істинно, то виконується тіло циклу, потім змінюються значення лічильників і знову перевіряється умова. Як тільки умова стає помилковим, цикл завершує свою роботу. У циклі for тіло циклу може жодного разу не виконуватися, якщо умова циклу помилково після ініціалізації, а може відбуватися зациклення, якщо умова завжди залишається справжнім. У нормальній ситуації тіло циклу виконується кінцеве число раз.Счетчікі циклу часто оголошуються безпосередньо в ініціалізатор і відпо венно є змінними, локалізованими в циклі, так що після завершення циклу вони перестають існувати. У тих випадках, коли передбачається можливість передчасного завершення циклу за допомогою одного з операторів переходу, лічильники оголошуються до циклу, що дозволяє аналізувати їх значення при виході з цікла.Цікли WhileЦікл while (вираз) є універсальним видом циклу, включених в усі мови програмування. Тіло циклу виконується до тих пір, поки залишається істинним вираз while. У мові C # у цього виду циклу дві модифікації - з перевіркою умови на початку циклу і в кінці циклу. Перша модифікація має наступний синтаксис: while (вираз) операторЕта модифікація циклу відповідає стратегії: «спочатку перевір, а потім роби». В результаті перевірки може виявитися, що і робити нічого не потрібно. Тіло такого циклу може жодного разу не виконуватися. Звичайно ж, можливо і зациклення. У нормальній ситуації кожне виконання тіла циклу - це черговий крок до завершення цікла.Цікл, перевіряючий умова завершення в кінці,

відповідає стратегії: «спочатку роби, а потім перевір». Тіло такого циклу виконується щонайменше один раз. Ось синтаксис цієї модифікації: do операторwhile (вираз); Цикл foreach Новим видом циклу, що не успадкованим від C ++, є цикл foreach, зручний при роботі з масивами, колекціями та іншими подібними контейнерами даних. Його синтаксис: foreach (тип ідентифікатор in контейнер) операторЦікл працює в повній відповідності зі своєю назвою - тіло циклу виконується для кожного елемента в контейнері. Тип ідентифікатора повинен бути узгоджений з типом елементів, що зберігаються в контейнері даних. Передбачається також, що елементи контейнера (масиву, колекції) впорядковані. На кожному кроці циклу ідентифікатор, що задає поточний елемент контейнера, отримує значення чергового елемента відповідно до порядку, встановленому на елементах контейнера. З цим поточним елементом і виконується тіло циклу. Тіло циклу виконується стільки разів, скільки елементів знаходиться в контейнері. Цикл закінчується, коли повністю перебрані всі елементи контейнера.Серьезним недоліком циклів foreach в мові C # є те, що цикл працює тільки на читання, але не на запис елементів. Так що наповнювати контейнер елементами доводиться за допомогою інших операторів циклу.