ВЫРАЖЕНИЯ И ОПЕРАТОРЫ ПРИСВАИВАНИЯ

СГУГиТ
СИБИРСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

Р. В. Гришин
mail-to: r.grishin54@gmail.com

СОДЕРЖАНИЕ ЛЕКЦИИ

- Введение
- Арифметические выражения
- Перегруженные операторы
- Преобразования типов
- Реляционные и булевы выражения
- Упрощенное вычисление
- Выражения присваивания
- Смешанный режим присваивания

СГУГиТ
СИБИРСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

ВВЕДЕНИЕ

- Выражения являются фундаментальным средством спецификации вычислений в языках программирования.
- Чтобы понять, как оценивать выражения, необходимо знать порядок оценки операторов и операндов.
- Суть императивных языков заключается в доминирующей роли операторов присваивания.

АРИФМЕТИЧЕСКИЕ ВЫРАЖЕНИЯ

Арифметические вычисления были одним из стимулов для разработки первых языков программирования

Арифметические выражения состоят из операторов, операндов, круглых скобок и вызовов функций

Вопросы проектирования арифметических выражений

- ❑ Правила старшинства операторов
- ❑ Правила ассоциативности операторов
- ❑ Порядок вычисления операндов
- ❑ Побочные эффекты вычисления операндов
- ❑ Перегрузка операторов
- ❑ Смешение типов в выражениях

ПРАВИЛА ПРИОРИТЕТА

Правила приоритета операторов для оценки выражений определяют порядок, в котором вычисляются «соседние» операторы с разным уровнем старшинства

Уровни приоритета:

- ❑ Круглые скобки
- ❑ Унарные операторы
- ❑ ** (если язык поддерживает)
- ❑ *, /
- ❑ +, -

ПРАВИЛА АССОЦИАЦИИ

Правила ассоциативности операторов для вычисления выражений определяют порядок, в котором вычисляются соседние операторы с одинаковым уровнем старшинства

Правила ассоциативности:

- Слева направо, за исключением **, который ассоциируется справа налево.
- Иногда унарные операторы ассоциируются справа налево (например, в FORTRAN).
- В APL все по-другому: все операторы имеют равный приоритет и все операторы ассоциируются справа налево
- Правила старшинства и ассоциативности можно отменить с помощью круглых скобок

УСЛОВНЫЕ ОПЕРАТОРЫ

С-подобные языки:

```
average = (count == 0) ?  0 : sum / count
```

Рассматривается так, как будто написано следующее:

```
if (count == 0)
    average = 0
else
    average = sum / count
```

СГУГиТ
СИБИРСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

ПОРЯДОК ВЫЧИСЛЕНИЯ

Переменные:

- ❑ Получение значение из памяти

Константы:

- ❑ Иногда извлекает значение из памяти
- ❑ Иногда константа находится внутри машинного языка

Выражения со скобками:

- ❑ Сначала вычисляются все операнды и операторы

СГУГиТ
СИБИРСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

ПОБОЧНЫЕ ЭФФЕКТЫ

Функциональные побочные эффекты: это когда функция изменяет двусторонний параметр или нелокальную переменную.

Проблема с функциональными побочными эффектами:

Когда функция, на которую ссылаются в выражении, изменяет другой операнд выражения; например, при изменении параметра.

```
a = 10;
/* assume that fun changes its parameter */
b = a + fun(&a);
```

СГУГиТ
СИБИРСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

ПЕРЕГРУЖЕННЫЕ ОПЕРАТОРЫ

- Использование оператора более чем для одной цели называется **перегрузкой оператора.**
- Некоторые из них являются общими (например, + для int и float).
- Некоторые представляют собой потенциальную проблему (например, & в C и C++).
  - Потеря возможности обнаружения ошибок компилятором (пропуск операнда должен быть обнаруживаемой ошибкой)
  - Некоторая потеря читабельности
  - Можно избежать путем введения новых символов (например, div в Паскале для целочисленного деления)
- C++ и Ada позволяют определять пользовательские перегруженные операторы
  - Пользователи могут определять бессмысленные операции
  - Читабельность может пострадать, даже если операторы имеют смысл

ПРЕОБРАЗОВАНИЯ ТИПОВ

Сужающее преобразование - преобразование объекта к типу, который не может включать все значения исходного типа

Пример: float к int

Расширяющееся преобразование - преобразование объекта в тип, который может включать по крайней мере приближенные значения всех значений исходного типа

Пример: int к float

ПРЕОБРАЗОВАНИЯ ТИПОВ

Смешанное выражение – это выражение, имеющее операнды разных типов.

Приведение типа – неявное преобразование типа.

Недостатки:

Приведение типов снижают способность компилятора обнаруживать ошибки типов.

В большинстве языков все числовые типы принудительно приводятся в выражениях расширяющих преобразований.

СГУГиТ
СИБИРСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

РЕЛЯЦИОННЫЕ ВЫРАЖЕНИЯ

Реляционный оператор - это оператор, который сравнивает значения двух операндов. Реляционное выражение состоит из двух операндов и одного реляционного оператора. Значение реляционного выражения - булево, за исключением случаев, когда булево не является типом, включенным в язык.

- Используемые символы операторов несколько отличаются в разных языках (!=, /=, .NE., <>, #).

БУЛЕВЫ ОПЕРАТОРЫ:

FORTRAN 77	*FORTRAN 90*	*C*	*Ada*
.AND.	and	&&	and
.OR.	or	||	or
.NOT.	not	!	not
			xor

УПРОЩЕННОЕ ВЫЧИСЛЕНИЕ

Упрощенное вычисление - это вычисление, при котором результат определяется без вычисления всех операндов и/или операторов.

(13 * a) * (b / 13 - 1)

- Если **a** является 0, нет необходимости вычислять **(b / 13 – 1)**

R = P && Q

- Если **P** ложно, то нет необходимости вычислять **Q**

УПРОЩЕННОЕ ВЫЧИСЛЕНИЕ

C, C++ и Java: используют упрощенные вычисления для обычных булевых операторов (&& и ||), но также предоставляют побитовые булевы операторы, которые не являются операторами с упрощенными вычислениями (& и |).

Все логические операторы в Ruby, Perl, ML, F#, Python вычисляются в упрощенном формате.

Проблема с отсутствием упрощенного вычисления:

```
index = 0;
while ( (index < length) && (LIST[index] != value) )
        index++;
```

- Когда index == length в LIST[index] возникнет проблема индексации

СГУГиТ
СИБИРСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

ОПЕРАТОРЫ ПРИСВАИВАНИЯ

Общий синтаксис:

<target_var> <assign_operator> <expression>

Оператор присваивания:

- = FORTRAN, BASIC, PL/I, C, C++, Java и тд.
- := ALGOL, Pascal, Ada

Оператор **=** может быть плохим решением, когда он перегружен реляционным оператором равенства

ТЕРНАРНЫЙ ОПЕРАТОР

Общий синтаксис (Perl):

```
($flag ? $total : %subtotal) = 0
```

Что эквивалентно:

```
if ($flag) {
  $total = 0
} else {
   $subtotal = 0
}
```

СОСТОВНЫЕ ОПЕРАТОРЫ

- Сокращенный способ указания часто используемой формы присвоения
- Введен в ALGOL; принят в C и языках, основанных на C.

a = a + b → a += b

- Унарные операторы присваивания в языках на базе C объединяют операции инкремента и декремента с присваиванием

sum = ++count
sum = count++

ПРИСВОЕНИЕ КАК ВЫРАЖЕНИЕ

- В языках на базе C, Perl и JavaScript оператор присваивания выдает результат и может быть использован в качестве операнда

```
while ( ( ch = getchar() ) != EOF) {…}
```

- Выполняется ch = getchar(); результат (присвоенный ch) используется в качестве условного значения для оператора while

Множественное присваивание:

```
($first, $second, $third) = (20, 30, 40);
```

СМЕШАННЫЕ ПРИСВАИВАНИЯ

- Операторы присваивания также могут иметь смешанный режим
- В Fortran, C, Perl и C++ переменной любого числового типа может быть присвоено значение любого числового типа.
- В Java и C# выполняются только расширяющие принуждения присваивания
- В Ada нет принудительного присваивания

«**Выражения и операторы присваивания** – ключевые элементы программирования, которые позволяют нам манипулировать данными и управлять потоком исполнения программы.»

Линус Торвальдс

СГУГиТ
СИБИРСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

Гришин Руслан Владимирович
mail to: r.grishin54@gmail.com