acm.mipt.ru

олимпиады по программированию на Физтехе

Раздел «Язык Си» . CfaqExpr :

Выражения

4.1

Q: Почему не работает код

```
a[i] = i++;
```

A: Подвыражение i++ приводит к побочному эффекту - значение i изменяется, что приводит к неопределенности, если i уже встречается в том же выражении. (Обратите внимание на то, что хотя в книге K&R говорится, что поведение подобных выражений не описано, стандарт ANSI/ISO утверждает, что поведение *не определено* - см. вопрос 5.23.)

См. ANSI Разд.3.3 с. 39.

4.2

Q: Пропустив код через свой компилятор, я получил на выходе 49. А разве, независимо от порядка вычислений, результат не должен быть равен 56?

```
int i = 7;
printf("%d\n", i++ * i++);
```

A: Хотя при использовании постфиксной формы операторов ++ и -- увеличение и уменьшение выполняется после того как первоначальное значение использовано, тайный смысл слова "после" часто понимается неверно. *Не* гарантируется, что увеличение или уменьшение будет выполнено немедленно после использования первоначального значения перед тем как будет вычислена любая другая часть выражения. Просто гарантируется, что измение будет произведено в какой-то момент до окончания вычисления (перед следующей "точкой последовательности" в терминах ANSI C). В приведенном примере компилятор умножил предыдущее значение само на себя и затем дважды увеличил i на 1.

Поведение кода, содержащего многочисленные двусмысленные побочные эффекты неопределено (см. вопрос 5.23). Даже не пытайтесь выяснить, как Ваш компилятор все это делает (в противоположность неумным упражнениям во многих книгах по C); в K&R мудро сказано: "Да хранит Вас Ваша невинность, если Вы не знаете, как это делается на разных машинах"

См.: K&R I Разд. 2.12 с. 50; K&R II Разд. 2.12 с. 54; ANSI Разд. 3.3 с. 39; CT&P Разд. 3.7 с. 47; PCS Разд. 9.5 с. 120-1. (Не принимайте во внимание H&S Разд. 7.12 с. 190-1, это устарело)

4.3

Q: Я экспериментировал с кодом

```
int i = 2;
i = i++;
```


Поиск

Поиск

Раздел «Язык Си»

Главная
Зачем учить С?
Определения

Инструменты:

Поиск
Изменения
Index
Статистика

Разделы

Информация
Алгоритмы
Язык Си
Язык Ruby
Язык Ассемблера
El Judge
Парадигмы
Образование
Сети
Objective C

Logon>>

Некоторые компиляторы выдавали i=2, некоторые 3, но один выдал 4. Я знаю, что поведение неопределено, но как можно получить 4?

A: Неопределенное (undefined) поведение означает, что может случиться *все* что угодно. См. вопрос 5.23.

4.4

Q: Люди твердят, что поведение неопределено, а я попробовал ANSI - компилятор и получил то, что ожидал.

A: Компилятор делает все, что ему заблагорассудится, когда встречается с неопределенным поведением (до некоторой степени это относится и к случаю зависимого от реализации и неописанного поведения). В частности, он может делать то, что Вы ожидаете. Неблагоразумно, однако, полагаться на это. См. также вопрос 5.18.

4.5

Q: Могу я использовать круглые скобки, чтобы обеспечить нужный мне порядок вычислений? Если нет, то разве приоритет операторов не обеспечивает этого?

A: Круглые скобки, как и приоритет операторов обеспечивают лишь частичный порядок при вычислении выражений. Рассмотрим выражение

```
f() + g() * h()
```

.

Хотя известно, что умножение будет выполнено раньше сложения, нельзя ничего сказать о том, какая из трех функций будет вызвана первой.

4.6

Q: Тогда как насчет операторов &&, ||, и запятой? Я имею в виду код типа

```
if((c = getchar()) == EOF || c == '\n')" ...
```

A: Для этих операторов, как и для оператора ?: существует специальное исключение; каждый из них подразумевает определенный порядок вычислений, т.е. гарантируется вычисление слева-направо. В любой книге по C эти вопросы должны быть ясно изложены.

См.: K&R I Разд. 2.6 с. 38, Разд. A7.11-12 с. 190-1; K&R II Разд. 2.6 с. 41, Разд. A7.14-15 с. 207-8; ANSI Разд. 3.3.13 с. 52, 3.3.14 с. 52, 3.3.15 с. 53, 3.3.17 с. 55, CT&P Разд. 3.7 с. 46-7.

4.7

Q: Если я не использую значение выражения, то как я должен увеличивать переменную i: так: ++i или так: i++ ?

A: Применение той или иной формы сказывается только на значении выражения, обе формы полностью эквивалентны, когда требуются только их побочные эффекты.

4.8

Q: Почему неправильно работает код

```
char a = 100, b = 100;
long int c = a * b;
```

A: Согласно общим правилам преобразования типов языка C, умножение выполняется с использованием целочисленной арифметики, и результат может привести к переполнению и/или усечен до того как будет присвоен стоящей слева переменной типа `long int`. Используйте явное приведение типов, чтобы включить арифметику длинных целых

```
long int c = (long int)a * b;
```

Заметьте, что код `(long int)(a * b)` *не* приведет к желаемому результату.

-- TatyanaDerbysheva - 06 Jan 2011