Неопределенное поведение

Побочные эффекты выражений

- Модификация данных.
- Обращение к переменным, объявленным как volatile.
- Вызов системной функции, которая производит побочные эффекты (например, файловый ввод или вывод).
- Вызов функций, выполняющих любое из вышеперечисленных действий.

Порядок вычисления выражений

```
i = 1;
x[i] = i++ + 1;

// Способ 1 (справа налево)
(i++ + 1) => 2, i = 2, x[2] = 2

// Способ 2 (слева направо)
x[1] = (i++ + 1) = 2, i = 2
```

Точки следования

- Компилятор вычисляет выражения. Выражения будут вычисляться почти в том же порядке, в котором они указаны в исходном коде: сверху вниз и слева направо.

- *Точка следования* – это точка в программе, в которой программист знает какие выражения (или подвыражения) уже вычислены, а какие выражения (или подвыражения) еще нет.

Точки следования

Определены следующие точки следования:

- Между вычислением левого и правого операндов в операциях &&, || и "," .

```
*p++ != 0 && *q++ != 0
```

- Между вычислением первого и второго или третьего операндов в тернарной операции.

```
a = (*p++) ? (*p++) : 0;
```

Точки следования

продолжение

- В конце полного выражения.

```
a = b;
if ()
switch ()
while ()
do{} while()
for ( x; y; z)
return x
```

Точки следования

продолжение

- Перед входом в вызываемую функцию.
  - Порядок, в котором вычисляются аргументы не определен, но эта точка следования гарантирует, что все ее побочные эффекты проявятся на момент входа в функцию.

- В объявлении с инициализацией на момент завершения вычисления инициализирующего значения.

```
int a = (1 + i++);
```

Порядок вычисления выражений

Почему результата выражения «x[i] = i++ + 1;» не определен?

- Порядок вычисления выражений и подвыражений между точками следования не определен.
- В выражении «x[i] = i++ + 1;» есть единственная точка следования, которая находится в конце полного выражения.
- В выражении «x[i] = i++ + 1;» есть два обращения к переменной i. Множественный доступ и является источником проблемы.

Избегайте сложных выражений!

Порядок вычисления выражений

Почему спецификация языка оставляет открытым вопрос, в каком порядке компиляторы должны вычислять выражения между точками следования?

Причина неопределенности порядка вычислений – простор для оптимизации.

Примеры

Виды неопределенного поведения

Unspecified behavior.

Стандарт предлагает несколько вариантов на выбор. Компилятор может реализовать любой вариант. При этом на вход компилятора подается корректная программа.

Например: все аргументы функции должны быть вычислены до вызова функции, но они могут быть вычислены в любом порядке.

Виды неопределенного поведения

Implementation-defined behavior.

Похоже на неспецифицированное (unspescified) поведение, но в документации к компилятору должно быть указано, какое именно поведение реализовано.

Например: результат x % y, где x и y целые, а y отрицательное, может быть как положительным, так и отрицательным.

Виды неопределенного поведения

Undefined behavior

Такое поведение возникает как следствие неправильно написанной программы или некорректных данных. Стандарт ничего не гарантирует, может случиться все что угодно.

Зачем нужно неопределенное поведение

Чтобы

- освободить разработчиков компиляторов от необходимости обнаруживать ошибки, которые трудно диагностировать.
- избежать предпочтения одной стратегии реализации другой.
- отметить области языка для расширения языка (language extension).

Примеры неопределенного поведения

- Использование неинициализированных переменных.
- Переполнение знаковых целых типов.
- Выход за границы массива.
- Использование «диких» указателей.

…

См. приложение J стандартна (стр. 490 - 502)

Примеры

Способы борьбы с неопределенным поведением

- Включайте все предупреждения компилятора, внимательно читайте их.
- Используйте возможности компилятора (-ftrapv).
- Используйте несколько компиляторов.
- Используйте статические анализаторы кода (например, clang).
- Используйте инструменты такие как valgrind, Doctor Memory и др.
- Используйте утверждения.