Точки следования, UB, sizeof, inline, ?:

Зацепин Михаил

Московский физико-технический институт

Москва, 2018

Затравка

```
i = ++i + i++;
```

Затравка

```
i = ++i + i++;

f(++i, ++i);
```

Затравка

```
i = ++i + i++;

f(++i, ++i);

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

UB и компания

- `ill-formed` - диагностика на этапе компиляции
- `implementation-defined behavior` - поведение зависит от реализации, но должно быть документировано (`std::size_t`)
- `unspecified behavior` - зависит от реализации, не обязано быть задокументировано, любой из вариантов валиден
- `undefined behavior` - нет ограничения на поведение программы (выход за пределы массива)

Side effects

- Модификация объектов
- Вызов функций ввода-вывода
- Вызов функций, которые модифицируют объект и т.д.

Точка следования (Sequence Point)

Точка следования - это точка последовательности выполнения, в которой все побочные эффекты от вычислений, стоящих раньше в последовательности, завершены, и никакие побочные эффекты, относящиеся к последующим вычислениям, не начали выполнятся.

Точка следования (Sequence Point)

Правила:

- Точка следования находится в конце каждого полного выражения (обычно, она расположена на точке с запятой)
- При вызове функции (включая `inline`) точка следования находится после вычисления всех аргументов функции (если таковые имеются), которое происходит перед выполнением любых выражений или инструкций в теле функции.
- Точка следования находится после копирования возвращаемого значения функции и перед выполнением любых выражений за пределами функции
- Выполнение функций не может чередоваться
- Операторы:

```
a && b
a || b
a ? b : c
a, b
```

Неопределенное поведение

Между точками следования, значение скалярного объекта не должно изменяться более одного раза

```
i = ++i + i++;
f(++i, ++i);
```

Неопределенное поведение

Между точками следования, значение скалярного объекта не должно изменяться более одного раза

```
i = ++i + i++;
f(++i, ++i);
```

Между точками следования, предыдущее значение скалярного объекта, которое модифицировано при вычислении выражения, должно быть доступно только для определения сохраняемого значения

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

Отношение sequenced-before

Вместо Sequence Points - отношение sequenced-before

```
a.b
a->b
a->*b
a(b1, b2, b3)
b @= a
a[b]
a << b
a >> b
```

Отношение sequenced-before

- Что-то осталось `Undefined`
- Поддержка C++17 компиляторами

Отношение sequenced-before

- Что-то осталось Undefined
- Поддержка C++17 компиляторами

```
void f()
{
  std::string s = "but I have heard it works even if you  d o n t
      believe in it";
  s.replace(0, 4, "").replace(s.find("even"), 4, "only").replace(s.
      find(" don't"), 6, "");
  assert(s == "I have heard it works only if you believe in it");
}
```

оператор sizeof

- Результат на этапе компиляции, то есть выражение не вычисляется!
- Возвращает значение типа `std::size_t`

Синтаксис:

- `sizeof( type )`
- `sizeof expression`

Размеры типов

- [unsigned|signed] sizeof(char) == 1

Размеры типов

- [unsigned|signed] sizeof(char) == 1
- Кол-во бит в char задается в константе CHAR_BIT. Обычно 8, но это неточно

Размеры типов

Type specifier	Equivalent type	Width in bits by data model				
		C++ standard	LP32	ILP32	LLP64	LP64
`short`	`short int`	at least 16	16	16	16	16
`short int`						
`signed short`						
`signed short int`						
`unsigned short`	`unsigned short int`					
`unsigned short int`						
`int`	`int`	at least 16	16	32	32	32
`signed`						
`signed int`						
`unsigned`	`unsigned int`					
`unsigned int`						
`long`	`long int`	at least 32	32	32	32	64
`long int`						
`signed long`						
`signed long int`						
`unsigned long`	`unsigned long int`					
`unsigned long int`						
`long long`	`long long int` (C++11)	at least 64	64	64	64	64
`long long int`						
`signed long long`						
`signed long long int`						
`unsigned long long`	`unsigned long long int` (C++11)					
`unsigned long long int`						

Размеры типов

Гарантируется лишь, что:

```
1 == sizeof(char) <= sizeof(short) <= sizeof(int) <= sizeof(long)
     <= sizeof(long long)
```

Размеры типов

Гарантируется лишь, что:

```
1 == sizeof(char) <= sizeof(short) <= sizeof(int) <= sizeof(long)
     <= sizeof(long long)
```

Поэтому возможна ситуация, когда все типы по 64 бита, и sizeof всегда возвращает 1