Выражения языка Си

Лекция 5

План лекции

- Выражения языка Си
  - Классы, приоритеты и ассоциативность операторов
  - Операторы, которые возвращают l-value
  - Порядок вычисления выражений
    - Точки следования
    - Побочные эффекты
  - Особенности выполнения операторов
    - Требования к операндам, значение и тип результата, побочные эффекты, well-defined, implementation specific, undefined behavior

Выражения языка Си

- Выражение – это последовательность операторов и операндов

- Выражение
  - Описывает вычисление значения, либо
  - Именует значение или функцию, либо
  - Имеет побочные эффекты, либо
    - Если имеет тип void, то ~~спец~~ побочные эффекты вероятнее всего
  - Делает «вот это всё» вместе

Какие бывают операторы 1/3

- Операторы делятся на классы по числу и расположению операндов

- Запись одного оператора состоит из одной или двух лексем

Класс	Число операндов	Положение оператора отн. операндов
Атомарные	0	
Префиксные	1	Перед
Постфиксные	1	После
Бинарные	2	Между
Тернарные	3	Между

Какие бывают операторы 2/3

- Операторы связываются с операндами по возрастанию своих приоритетов
- Приоритеты задаются целыми числами

П(*)	П(+)	Возможная расстановка скобок в x*x+y*y
13	12	(x*x)+(y*y)
12	13	(x*(x+y))*y, x*((x+y)*y)
12	12	((x*x)+y)*y (x*(x+y))*y (x*x)+(y*y) x*(x+(y*y)) x*((x+y)*y)

Какие бывают операторы 3/3

- Ассоциативность бинарных операторов задает расстановку скобок в выражениях, содержащих операторы одного приоритета
- Левоассоциативные -- слева направо
- Правоассоциативные -- справа налево
- Операторы языка Си одного приорит ассоциативность
  - Иначе расстановка скобок неоднозначна

A(-)	A(+)	Расстановка скобок в x-x+y-y
л	л	((x-x)+y)-y
п	п	x-(x+(y-y))

- Выражения языка Си
  - Классы, приоритеты и ассоциативность операторов
  - Операторы, которые возвращают l-value
  - Порядок вычисления выражений
    - Точки следования
    - Побочные эффекты
  - Особенности исполнение операторов

Зачем нужны l-value?

- Значения, которым гарантировано соответствует участок памяти, называются l-value
- Только 5 операторов в языке Си возвращают l-value – см. следующий слайд
- Остальные операторы возвращают обычные значения
  - Место для хранения этих значений (память или регистры процессора) выбирает компилятор
- Прагматика l-value
  - Придание точного смысла операторам, использующим адреса памяти

```
&A[i] // OK
&(A[i] + A[j]) // не OK
A[i] = 5 // OK
A[i] + A[j] = 5  // не OK
```

  - Больше свободы компилятору при оптимизации объектного кода

Операторы, которые возвращают l-value

- l-value получаются при выполнении операторов
  - Доступ к значению переменной
  - Доступ через указатель *
  - Доступ к элементу массива a[k]
  - Доступ к полю структуры или объединения student.name
  - Доступ к полю структуры или объединения через указатель student->name

- Все остальные операторы возвращают обычные значения

Операторы, которые требуют l-value

- Левый операнд во всех видах присваивания =, += и т.п.
- Взятие адреса &
- Префиксные и постфиксные ++ и --

Операторы, которые возвращают l-value

- Пример 1

```
int x;
x = 2; // x – l-value
int A[10];
A[5] = 5+x; // A[5] – l-value, 5+x – не l-value
```

- Пример 2

```
int x, y;
(x < y ? x : y) = 1;//  ошибка, т.к. (x < y ? x : y) не l-value
*(x < y ? &x : &y) = 1; // ОК, т.к. *(x < y ? &x : &y) – l-value
```

- Пример 3

```
(A[i] < A[j] ? A[i] : A[j]) = 1; // ошибка
A[ A[i] < A[j] ? i : j ] = 1; // ОК
```

Точки следования, побочные эффекты

- Побочный эффект вычисления выражения – это факт изменения содержимого ячеек памяти в процессе вычисления выражения
  - Присваивание
    - x = 1;
  - Сложный побочный эффект
    - i = 0; A[i++] = i++; // чему равно i – 0 или 1?
    - В каком порядке выполнятся = и ++?
    - Определён ли вообще порядок исполнения = и ++?

Точки следования, побочные эффекты

- Точка следования (sequence point) -- точка программы, в которой гарантируется, что все побочные эффекты предыдущих вычислений уже проявились, а побочные эффекты последующих ещё отсутствуют

Точки следования, побочные эффекты

1. Между вычислением левого и правого операндов в операциях &&, || и , (запятая)
2. Между вычислением первого и второго или третьего операндов в операции ?:
3. В конце всего выражения
4. Перед входом в вызываемую функцию
5. В объявлении с инициализацией на момент завершения вычисления инициализирующего значения
6. В остальном порядок выполнения операций определяет компилятор

Точки следования, побочные эффекты

- Пример 1
```
while (*p++ != 0 && *q++ != 0) *p = *q;
```

- Побочный эффект *p++ != 0 проявится до начала вычисления *q++ != 0 -- Правило 1
- Побочный эффект *q++ != 0 проявится до начала вычисления *p = *q -- Правило 3
- Никогда не пишите так =)

А что делают эти циклы?

```
while (*p != 0 && *q != 0) *p++ = *q++;
while (*q != 0) *p++ = *q++;
```

Точки следования, побочные эффекты

- Пример 2

```
int A[3] = { 1, 0, 2 }, *p = A;
int a = (*p++) ? (*p++) : 0; // чему равно a?
```

- Точка следования находится после первого *p++
- p уже увеличена на 1 при вычислении второго *p++
- Никогда, никогда не пишите так!

Точки следования, побочные эффекты

- Пример 3

```
int i = 0, j = i++, k = i++; // (1)
int x = f(i++) + g(j++) + h(k++); // (2)
```

- Каждая из переменных i, j и k принимает новое значение перед входом в f, g и h соответственно, но при этом…
  - Не определен
    - Порядок вызова функций f(), g(), h() и порядок инкрементов i, j, k в строке 2
  - Если i, j и k – глобальные переменные, то не определены
    - Значения j и k внутри f
    - Значения i и k внутри g
    - Значения i и j внутри h

Приоритеты операторов в языке Си

Лексемы	Оператор	Класс	Приор-т	Ассоц-ность
Переменные Константы	Доступ к значению константы или переменной	атомарный	16	нет
a[k]	Доступ к элементу массива	постфиксный	16	слева направо
f(…)	Вызов функции	постфиксный	16	слева направо
.	Доступ к элементу struct или union	постфиксный	16	слева направо
->	Доступ к элементу struct или union через указатель	постфиксный	16	слева направо
k++ k--	Доступ к значению k и послед. увеличение или уменьшение k на 1	постфиксный	16	слева направо
++k --k	Увеличение или уменьшение k на 1 и послед. доступ к полученному значению k	префиксный	15	справа налево
sizeof	Размер значения или типа в байтах	префиксный	15	справа налево
~	Побитовое НЕ	префиксный	15	справа налево
!	Логическое НЕ	префиксный	15	справа налево

Приоритеты операторов в языке Си

Лексемы	Оператор	Класс	Приор-т	Ассоциативность
*	Доступ через указатель	префиксный	15	справа налево
(имя типа)	Преобразование типа	префиксный	14	справа налево
* / %	Умножение, деление, остаток от деления	бинарный	13	слева направо
+ -	Сложение, вычитание чисел и указателей	бинарный	12	слева направо
<< >>	Сдвиг влево или вправо в 2 с.с.	бинарный	11	слева направо
< > <= >=	Сравнение чисел и указателей	бинарный	10	слева направо
== !=	Проверка равенства и различия	бинарный	9	слева направо
&	Побитовое И	бинарный	8	слева направо
^	Побитовое исключающее ИЛИ	бинарный	7	слева направо
|	Побитовое ИЛИ	бинарный	6	слева направо
&&	Логическое И	бинарный	5	слева направо
||	Логическое ИЛИ	бинарный	4	слева направо
c ? в1 : в2	в1 (если c != 0) или в2 (если c == 0)	тернарный	3	справа налево

Приоритеты операторов в языке Си

Лексемы	Оператор	Класс	Приор-т	Ассоц-ность
= += -= *= /= %= <<= >>= &= ^= |=	Вычисление правого операнда и послед. запись полученного значения в ячеку памяти, определяемую левым операндом (присваивание)	бинарный	2	справа налево
,	Последовательное вычисление операндов	бинарный	1	слева направо

Операторы языка Си

Требования к виду и типам операндов	
Правило определения типа результата	
Правило вычисления результата	
Побочные эффекты, кроме побочных эффектов при вычислении операндов	Как меняется состояние памяти в результате исполнения самой операции
Условия well defined	Когда результат зависит только от операндов
Условия implementation defined	Когда результат зависит от операндов и компилятора
Условия undefined behavior	Когда результат зависит от операндов, компилятора и стечения обстоятельств («фазы луны», «флагов компиляции» и т.п.)

Первичные выражения

Требования к виду и типам операндов	Идентификатор, явная константа, строковый литерал или (выр)
Правило определения типа результата	Переменная -> по описанию; константа -> по записи; литерал, функция -> указатель; (выр) -> тип выр
Правило вычисления результата	Переменная -> читаем из памяти во время исполнения; константа -> по записи во время компиляции; литерал, функция -> во время линковки; (выр) -> вычисляем выр
Побочные эффекты	Нет
Условия well defined	Константы, литералы, функции, (выр) -> всегда; переменная -> если присвоено значение
Условия implementation specific	Вещественные константы, записанные в 10 с.с. и неточно представимые в 2 с.с.
Условия undefined behavior	Переменная -> если не присвоено значение

Доступ к элементу массива A[k]

Требования к виду и типам операндов	Выражение A имеет тип T*, выражение k имеет целочисленный тип
Правило определения типа результата	Тип T
Правило вычисления результата	Значение, начиная с адреса A + k*размер(T) Очередность вычисления A и k не определена
Побочные эффекты	Нет
Условия well defined	Память по адресам [A + размер(T)*k, A + размер(T)*(k+1) - 1] доступна и ей присвоено значение, адрес A кратен размер(T)
Условия implementation specific	
Условия undefined behavior	Нарушены условия well defined

Вызов функции f(…)

Требования к виду и типам операндов	Тип выражения f – функция или указатель на функцию, типы фактических параметров соответствуют формальным параметрам в описании f; см. лекцию 5 про функции
Правило определения типа результата	Описание f
Правило вычисления результата	Вычисление значений фактических параметров Вычисление выражения f Исполнение тела f Очередность вычисления фактических параметров и выражения f не определена
Побочные эффекты	Создание «стекового кадра» на время работы тела f для хранения локальных переменных из тела f, результата f и адреса возврата из f
Условия well defined	Достаточно памяти для создания стекового кадра; одинаковый формат стекового кадра у вызывающего и у вызываемого; память адресу f
Условия implementation specific	Нет
Условия undefined behavior	Нарушено well defined

Доступ к элементу struct или union s.x

Требования к виду и типам операндов	Пусть T -- тип выражения s. T – struct или union, x – имя элемента в T; см. лекцию 7 про структуры
Правило определения типа результата	Тип элемента с именем x в T
Правило вычисления результата	Значение элемента с именем x в значении выражения s
Побочные эффекты	Нет
Условия well defined	Элементу с именем x в значении выражения s присвоено значение
Условия implementation specific	Нет
Условия undefined behavior	Нарушено well defined

Доступ к элементу struct или union s->x

Требования к виду и типам операндов	Пусть T* -- тип выражения s. T – struct или union, x – имя элемента в T; см. лекцию 7 про структуры
Правило определения типа результата	Тип элемента с именем x в T
Правило вычисления результата	Значение элемента с именем x в значении, хранящемся под адресу, равному значению выражения s
Побочные эффекты	Нет
Условия well defined	Элементу с именем x в значении, хранящемся по адресу, равному значению выражения s, присвоено значение
Условия implementation specific	Нет
Условия undefined behavior	Нарушено well defined

Постфиксный инкремент/декремент k++, k--

Требования к виду и типам операндов	Тип выражения k целый или указатель, является l-value – см. эту лекцию
Правило определения типа результата	Тип выражения k
Правило вычисления результата	k++, k-- ---> k
Побочные эффекты	k++ ---> k = k + 1, k-- ---> k = k - 1 в некоторый момент после вычисления значения и ближайшей точкой следования
Условия well defined	Память, хранящая значение выражения k, доступна для чтения и записи и присвоено значение
Условия implementation specific	
Условия undefined behavior	Нарушено well defined

Префиксный инкремент/декремент ++k, --k

Требования к виду и типам операндов	Тип выражения k целый или указатель, является l-value – см. эту лекцию
Правило определения типа результата	Тип выражения k
Правило вычисления результата	++k --> k + 1, --k --> k – 1
Побочные эффекты	++k --> k = k + 1, --k --> k = k - 1 к моменту вычисления значения выражения
Условия well defined	Память, хранящая значение выражения k, доступна для чтения и записи и присвоено значение
Условия implementation specific	
Условия undefined behavior	Нарушено well defined

Размер значения или типа sizeof x

Требования к виду и типам операндов	x -- выражение или конструкция вида (*абстрактный-объявитель*)
Правило определения типа результата	size_t из stddef.h
Правило вычисления результата	во время компиляции; размер памяти в байтах, занимаемый значениями типа, который имеет выражение x или абстрактный объявитель; int* x = malloc(sizeof(*x)); // OK
Побочные эффекты	Нет
Условия well defined	
Условия implementation specific	Всегда
Условия undefined behavior	

Побитовое НЕ ~x

Требования к виду и типам операндов	Выражение x имеет целочисленный тип
Правило определения типа результата	Тип выражения x
Правило вычисления результата	1 --> 0, 0 --> 1 для всех битов в значении выражения x, включая незначащие нули
Побочные эффекты	Нет
Условия well defined	Всегда
Условия implementation specific	
Условия undefined behavior	

Логическое НЕ !x

Требования к виду и типам операндов	Выражение x имеет целочисленный тип
Правило определения типа результата	int
Правило вычисления результата	x = 0 --> 1, x != 0 --> 0
Побочные эффекты	Нет
Условия well defined	Всегда
Условия implementation specific	
Условия undefined behavior	

Смена/сохранение знака числа -x и +x

Требования к виду и типам операндов	Выражение x имеет числовой тип
Правило определения типа результата	Тип выражения x
Правило вычисления результата	-x --> - значение выражения x; +x --> значение выражения x
Побочные эффекты	Нет
Условия well defined	Всегда
Условия implementation specific	
Условия undefined behavior	

Взятие адреса &x

Требования к виду и типам операндов	Выражение x является l-value
Правило определения типа результата	T*, где T – тип выражения x
Правило вычисления результата	Адрес, по которому хранится значение выражения x
Побочные эффекты	Нет
Условия well defined	
Условия implementation specific	Всегда
Условия undefined behavior	

Доступ через указатель *x

Требования к виду и типам операндов	Выражение x имеет тип T*
Правило определения типа результата	T
Правило вычисления результата	Значение, хранящееся по адресу, равному значению выражения x
Побочные эффекты	Нет
Условия well defined	Память по адресу, равному значению выражения x, доступна для чтения и присвоено значение
Условия implementation specific	
Условия undefined behavior	Не well defined

Преобразование типа (T) x

Требования к виду и типам операндов	Скалярный тип – либо простой тип, либо тип функции, либо указатель, либо enum. Пусть Y – тип выражения x. Если Y – вещественный, то T – простой тип или enum; иначе если Y --скалярный и невещественный, то T – любой скалярный; иначе если T – void, то Y любой; иначе T = Y
Правило определения типа результата	T
Правило вычисления результата	См. правила явных преобразований типов в пред. лекциях
Побочные эффекты	Нет
Условия well defined	(Один из T, Y -- не указатель или T = TT*, Y = YY* – указатели и размер YY кратен размеру TT) И (нет преобразования указателя в целое)
Условия implementation specific	
Условия undefined behavior	

Умножение, деление, остаток x op y, op = */%

Требования к виду и типам операндов	Выражения x и y имеют числовой тип; x % y --> x и y имеют целочисленный тип
Правило определения типа результата	См. правила неявных преобразований типов
Правило вычисления результата	Пусть T -- тип(x op y). Если x или y вещественные, то (T)x op (T)y; иначе (T)((T)x op (T)y). Очерёдность вычисления x и y не определена
Побочные эффекты	Нет
Условия well defined	Не возникает переполнения; x % y --> x >= 0, y > 0; x / y --> y != 0
Условия implementation specific	Переполнение; x % y и x либо y < 0; x / y и x != 0 и y = 0
Условия undefined behavior	x % 0, x / 0

Сложение, вычитание x op y, op = +-

Требования к виду и типам операндов	Типы выражений x и y числовые, или один из них целочисленный, а второй – указатель, op = - и оба указатели
Правило определения типа результата	Если типы выражений числовые, то как для */%; иначе см. лекцию 6 про указатели
Правило вычисления результата	Если типы выражений числовые, то как для */%; иначе указатель
Побочные эффекты	Нет
Условия well defined	Типы выражений числовые и не возникает переполнения; для указателей см. лекцию 6
Условия implementation specific	Типы выражений числовые и возникает переполнение; для указателей см. лекцию 6
Условия undefined behavior	Нет

Сдвиг x op y, op = << >>

Требования к виду и типам операндов	Выражения x и y имеют целочисленный тип
Правило определения типа результата	размер(x op y) = MAX(размер(x), размер(int)) беззнака(x op y) = беззнака(x)
Правило вычисления результата	значение x >> y = значение x / $2^{\text{значение y}}$, значение x << y = (значение x * $2^{\text{значение y}}$) mod $2^{8*\text{размер(x << y)}}$ Очерёдность вычисления x и y не определена
Побочные эффекты	Нет
Условия well defined	op = >> и значение x >= 0; op = << и тип x без знака; op = <<, тип x со знаком, значение x >= 0 и значение x << y представимо
Условия implementation specific	op = >>, значение x < 0, значение y от 0 до 8*размер(x op y) – 1
Условия undefined behavior	значение y < 0 или >= 8*размер(x op y)

Сравнение x op y, op = < > <= >=

Требования к виду и типам операндов	Типы x и y скалярные, и если один из них указатель, то другой не вещественный
Правило определения типа результата	int
Правило вычисления результата	Пусть T – наименьший тип, к которому преобразуются тип x и тип y. Если для значений (T)x и (T)y выполнено op, то 1; иначе 0 Очерёдность вычисления x и y не определена
Побочные эффекты	Нет
Условия well defined	тип x и тип y числовые; тип x и тип y указатели на элементы одного struct, union, или массива + один элемент за концом массива
Условия implementation specific	один из типов целочисленный, второй – указатель
Условия undefined behavior	

Проверка равенства x op y, op = == !=

	см. < > <= >=
Требования к виду и типам операндов	Типы x и y скалярные, и если один из них указатель, то другой не вещественный
Правило определения типа результата	int
Правило вычисления результата	Пусть T – наименьший тип, к которому преобразуются тип x и тип y. Если для значений (T)x и (T)y выполнено op, то 1; иначе 0 Очерёдность вычисления x и y не определена
Побочные эффекты	Нет
Условия well defined	тип x и тип y числовые; тип x и тип y указатели на элементы одного struct, union, или массива + один элемент за концом массива
Условия implementation specific	один из типов целочисленный, второй – указатель
Условия undefined behavior	Типы x и y скалярные, и если один из них указатель, то другой не вещественный

Побитовое И, ИсклИЛИ, ИЛИ x op y, op = &^|

Требования к виду и типам операндов	Типы выражений x и y целочисленные
Правило определения типа результата	размер(x op y) = MAX(размер(x), размер(y)) беззнака(x op y) = беззнака(x) ИЛИ беззнака(y)

Правило вычисления результата	бит значения x op y вычисляется по соответствующим битам значений x и y, включая незначащие нули
	`& 0 1  ^ 0 1  | 0 1`
	`0 0 0  0 0 1  0 0 1`
	`1 0 1  1 1 0  1 1 1`
	Очерёдность вычисления x и y не определена

Побочные эффекты	Нет
Условия well defined	Всегда
Условия implementation specific	
Условия undefined behavior	

Логические И и ИЛИ x op y, op = && ||

Требования к виду и типам операндов	Типы выражений x и y скалярные
Правило определения типа результата	int
Правило вычисления результата	op = &&: если значение !!(int)x == 0, то 0; иначе значение !!(int)y op = ||: если значение !!(int)x == 1, то 1; иначе значение !!(int)y Первым вычисляется x и потом, возможно, y
Побочные эффекты	Нет
Условия well defined	если x или y указатель, то он преобразуется в int и обратно с сохранением значения
Условия implementation specific	
Условия undefined behavior	x или y указатель, и его значение не сохраняется при преобразовании в int и обратно

Условное выражение с ? e1 : e2

Требования к виду и типам операндов	Тип выражения с скалярный, и одно из условий: тип выражений e1 и e2 -- void, или типы обоих выражений -- не void и они преобразуются друг к другу с помощью преобразования типов
Правило определения типа результата	если оба типа void, то void; иначе …
Правило вычисления результата	Если значение с != 0, то значение e1; иначе значение e2
Побочные эффекты	Нет
Условия well defined	Всегда, кроме implementation specific
Условия implementation specific	Значение e1 или e2 преобразуется из указателя в целое
Условия undefined behavior	

Присваивание x op y, op = = += -= *= …

Требования к виду и типам операндов	Тип y может быть преобразован к типу x; x является l-value
Правило определения типа результата	Тип x
Правило вычисления результата	op = op1= --> (значение выражения x) op1 (значение выражения y, преобразованное к типу x)
Побочные эффекты	Память по адресу, равному адресу значения x, заменяется на результат
Условия well defined	Всегда кроме implementation specific и undefined behavior
Условия implementation specific	При вычислении результата указатель преобразуется в целое
Условия undefined behavior	Память по адресу значения x не доступна для чтения и записи

Последовательное вычисление x , y

Требования к виду и типам операндов	
Правило определения типа результата	Тип y
Правило вычисления результата	Вычисляем значение x; результат = значение y
Побочные эффекты	Нет
Условия well defined	Всегда
Условия implementation specific	
Условия undefined behavior	

Заключение

- Классы, приоритеты и ассоциативность операторов
- Операторы, которые возвращают l-value
- Выражения l-value
- Порядок вычисления выражений, точки следования, побочные эффекты