Выражения языка Си

А. А. Вылиток

1. Объекты и *l*-обозначения

Объект — это область памяти, в которую можно записывать определенную информацию и считывать ее оттуда. *l-обозначение* — это выражение, означающее объект. Термин «*l*-обозначение» произошел от записи присваивания $E_1 = E_2$, в которой левый (*left* — левый (англ.), отсюда и буква *l*) операнд E_1 должен быть *l*-обозначением.

Примеры:

Выражение	Комментарий
int i;	*i* — переменная целого типа *int*
*int *pi = &i*;	*pi* — переменная типа «указатель на *int*»
i = 10;	выражение *i* здесь является *l*-обозначением, оно обозначает объект целого типа
**pi = i* + 1;	выражение **pi* обозначает тот же объект целого типа

Некоторые операции могут образовывать *l*-обозначения, другие — нет.

Не могут быть *l*-обозначениями:

- имена функций
- константы перечислимых типов
- вызовы функций
- операции приведения типов

Могут быть *l*-обозначениями:

- имена переменных
- выражения в скобках вида: (*e*), где *e* является *l*-обозначением
- *e*[*k*]
- *e.имя*, если *e* является *l*-обозначением
- *e−>имя*
- **e*
- строковая константа

По способу записи все операции подразделяются на классы:

- постфиксные: ⟨*операнд*⟩ ⟨*операция*⟩ (например, *X*++)
- префиксные: ⟨*операция*⟩ ⟨*операнд*⟩ (например, ++*X*)
- инфиксные: ⟨*операнд*⟩ ⟨*операция*⟩ ⟨*операнд*⟩ (например, *X* + *Y*)

2. Приоритет и ассоциативность операций

Каждая операция имеет вполне определенный *приоритет* и *ассоциативность*. Это позволяет без лишних скобок однозначно понимать, к какой операции относится операнд, стоящий между двумя соседними операциями. Например, в выражении

a + *b* * *c*

операнд *b* относится к операции умножения, так как у нее приоритет выше, чем у сложения. Поэтому эквивалентное выражение со скобками будет таким:

$$a + (b * c)$$

Операции одинакового приоритета либо *левоассоциативны*, либо *правоассоциативны*. Это значит, что операнд, находящийся между двумя такими операциями, относится либо к левой операции, либо (в случае правой ассоциативности) к правой. В языке Си операции сложения и вычитания имеют одинаковый приоритет и левую ассоциативность. Поэтому в выражении

$$a - b + c$$

операнд *b* относится к левой операции, т.е. к «минусу», а не к «плюсу», и это выражение эквивалентно следующему выражению со скобками:

$$((a - b) + c)$$

Таким образом, левоассоциативные операции группируются с помощью скобок **слева направо**; правая ассоциативность задаёт способ группировки операций **справа налево**. Примером правоассоциативной операции в языке Си является операция присваивания. Выражение с операциями присваивания

$$a = b = c$$

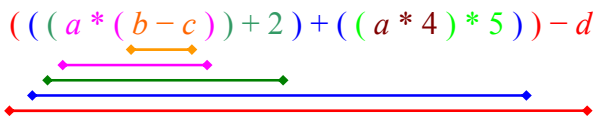
эквивалентно выражению

$$a = (b = c)$$

Приоритет и ассоциативность позволяют установить границы подвыражений, являющихся операндами, и заключить их в скобки. Например, если в выражении

$$a*(b - c) + 2 + a*4*5 - d$$

каждый нетривиальный операнд заключить в скобки, получим:



Каждая операция вместе со своими операндами выделена в этом примере отдельным цветом.

Порядок выполнения операций зависит не только от расстановки скобок, но и от порядка вычисления операндов (об этом ниже в п. 4).

В языке Си все постфиксные операции имеют одинаковый приоритет — самый высокий по отношению к другим операциям, — и естественным образом группируются слева направо. Приоритет префиксных операций ниже, чем у постфиксных, но выше, чем у инфиксных. Префиксные операции естественным образом группируются справа налево. Например, в выражении

$$\&*p[i]{+}{+}$$

операции & и * — префиксные, [] и ++ — постфиксные. Поэтому скобки будут расставлены так:

$$\&(*((p[i])++))$$

Инфиксные операции разбиты на несколько групп, имеющих разные приоритеты и ассоциативность (см. таблицу в п. 6).

3. Основное значение операции и побочные эффекты

Каждая операция имеет операнды определенных типов и задает способ получения по значениям этих операндов нового значения определенного типа.

Некоторые операции имеют *побочный эффект*. При выполнении этих операций, кроме основного эффекта — вычисления значения — происходят изменения объектов или файлов. Таковы, например, операции присваивания. Значением выражения $a = b$ будет значение переменной b, приведенное к типу переменной a, и в качестве побочного эффекта это значение будет присвоено переменной a.

Иногда выражения вычисляются только ради побочных эффектов, вычисленное значение игнорируется. Вызовы функций и операция приведения типа могут давать «пустое» значение, т.е. значение типа *void*.

4. Порядок вычисления операндов

В некоторых операциях порядок вычисления операндов строго фиксирован. Например, в логическом «ИЛИ» сначала вычисляется левый операнд, затем — правый. В большинстве операций языка Си порядок вычисления операндов не определен. Это значит, что в реализации может быть выбран любой порядок вычислений, и даже перемежаться правый и левый порядок одновременно.

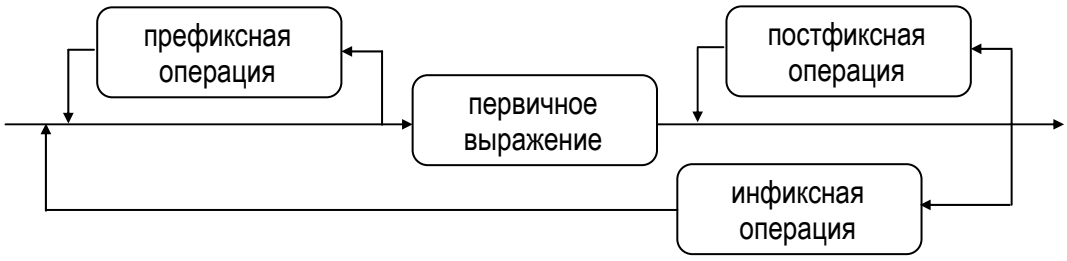
5. Синтаксис выражений

Выражения строятся из *первичных выражений* и знаков операций. Первичные выражения вычисляются в первую очередь, они имеют наивысший приоритет. К первичным выражениям относятся:

- идентификатор
- лексическая константа
- выражение в скобках
- *sizeof* (⟨*описание типа*⟩)

Синтаксис выражений задается диаграммой:

⟨*выражение*⟩ ::=



6. Таблица приоритетов операций

В таблице перечислены все операции языка Си в порядке убывания приоритетов. Высший приоритет имеет номер 16, низший — номер 1. Для каждой операции указаны способ записи, класс и ассоциативность.

Лексемы	Название	Нотация	Класс	Прио-ритет	Группировка
[…]	индекс	*X* [*Y*]	постфикс.	16	слева направо
(…)	вызов функции	*X* (*Y*)			
. ⟨*имя*⟩	прямой выбор члена ⟨имя⟩	*X*.⟨*имя*⟩			
–>⟨*имя*⟩	выбор члена ⟨имя⟩ через указатель	*X*–>⟨*имя*⟩			
++	увеличение	*X*++			
––	уменьшение	*X*––			
sizeof	размер	*sizeof X*	префикс.	15	справа налево
++	увеличение	++*X*			
––	уменьшение	––*X*			
&	адрес	&*X*			
*	разыменование	**X*			
+	плюс	+*X*			
–	минус	–*X*			
~	побитовое НЕ	~*X*			
!	логическое НЕ	!*X*			
(⟨*тип*⟩)	приведение типа	(⟨*тип*⟩) *X*	префикс.	14	справа налево
*	умножение	*X* * *Y*	инфиксн.	13	слева направо
/	деление	*X* / *Y*			
%	остаток	*X* % *Y*			
+	сложение	*X* + *Y*	инфиксн.	12	слева направо
–	вычитание	*X* – *Y*			
<<	сдвиг влево	*X* << *Y*	инфиксн.	11	слева направо
>>	сдвиг вправо	*X* >> *Y*			
<	меньше	*X* < *Y*	инфиксн.	10	слева направо
<=	меньше или равно	*X* <= *Y*			
>	больше	*X* > *Y*			
>=	больше или равно	*X* >= *Y*			
==	равно	*X* == *Y*	инфиксн.	9	слева направо
!=	не равно	*X* != *Y*			
&	побитовое И	*X* & *Y*	инфиксн.	8	слева направо
^	побитовое	*X* ^ *Y*	инфиксн.	7	слева направо

Лексемы	Название	Нотация	Класс	Прио-ритет	Группировка
	исключающее ИЛИ				
|	побитовое ИЛИ	*X* | *Y*	инфиксн.	6	слева направо
&&	логическое И	*X* && *Y*	инфиксн.	5	слева направо
||	логическое ИЛИ	*X* || *Y*	инфиксн.	4	слева направо
? :	условие	*Z* ? *X* : *Y*	инфиксн.	3	справа налево
=	присваивание	*X* = *Y*	инфиксн.	2	справа налево
*=	присваивание с умножением	*X* *= *Y*			
/=	присваивание с делением	*X* /= *Y*			
%=	присваивание с остатком	*X* %= *Y*			
+=	присваивание со сложением	*X* += *Y*			
−=	присваивание с вычитанием	*X* −= *Y*			
<<=	присваивание со сдвигом влево	*X* <<= *Y*			
>>=	присваивание со сдвигом вправо	*X* >>= *Y*			
&=	присваивание с побитовым И	*X* &= *Y*			
^=	присваивание с побитовым исключающим ИЛИ	*X* ^= *Y*			
|=	присваивание с побитовым ИЛИ	*X* |= *Y*			
,	запятая	*X*, *Y*	инфиксн.	1	слева направо

7. Виды скобок в выражениях

В выражениях языка Си встречаются три вида скобок:

1) ( )
2) [ ]
3) ? :

Выражение должно быть сбалансировано по всем видам скобок. Подвыражения, ограниченные парами скобок любых видов, могут вкладываться друг в друга. Два подвыражения, ограниченные скобками, либо вложены одно в другое, либо не пересекаются.

Примеры:

Выражение	Комментарий
f (*a**(*b*+*c*) ? −*s*+*m*[*x* ? *y*−1 : *z*] : *d*) * (5−*a*)	правильное выражение
(*m*[*j*)+*i*]	не является выражением

Круглые скобки имеют четыре различные роли:

- первичное выражение | (a + b)
- вызов функции | *f* (*x*, *y*)
- приведение типа | (*int*) (5.2+1)
- *sizeof*(⟨*mun*⟩) | *sizeof* (*int* *)

Роль квадратных скобок — индексирование элементов массива. Кроме того, квадратные скобки используются при описании массива (в них может указываться число элементов в массиве). В выражении *X* [*Y*] запись «[*Y*]» считается постфиксной операцией, применяемой к операнду «*X*». Также и в вызове функции *X* (*Y*), конструкция «(*Y*)», где *Y* — список аргументов (возможно, пустой) считается постфиксной операцией по отношению к операнду «*X*», именующему функцию.

Скобки ? : ограничивают в условной операции *Z* ? *X* : *Y* средний операнд *X*. Конструкция «? *X* :» по отношению к левому (*Z*) и правому (*Y*) операндам рассматривается как инфиксная операция, имеющая правую ассоциативность и приоритет, который ниже, чем у логического ИЛИ, но выше, чем у операций присваивания.

Примеры:

Выражение	Эквивалентное выражение
a ? *b* : *c* ? *d* : *e*	*a* ? *b* : (*c* ? *d* : *e*)
x = *y* = *a* ? *b* + *c* : *d* || *e* ? *b* − *c* : *c***b*	*x* = (*y* = (*a* ? (*b*+*c*) : ((*d* || *e*)?(*b*−*c*) : (*c***b*))))
a ? *b* ? *c* : *d* ? *e* : *f* : *g*	*a* ? (*b* ? *c* : (*d* ? *e* : *f*)) : *g*

Запись *x* = *a* ? *b* : *d* = *y* не является правильным выражением, поскольку в соответствии с приоритетами она трактуется как *x* = (*a* ? *b* : *d*) = *y*, но слева от правой операции присваивания должно быть выражение, которое являлось бы *l*-обозначением.

8. Точки последовательных вычислений

Побочный эффект при вычислении выражения — это занесение в память значений объектов, изменение состояния файла либо доступ к *volatile*-объектам. К побочным эффектам могут приводить вызовы функций, если вызываемая функция изменяет «внешний» по отношению к ней объект. Кроме того, операции увеличения и уменьшения (префиксные и постфиксные) а также все операции присваивания, помимо своего основного значения, обладают побочным эффектом — изменяют объекты. Обычно эти операции и применяют ради изменения объектов. Однако, побочный эффект не обязан «проявиться» одновременно с вычислением значения операции. Например, *y* = **p*++; может быть вычислено как

temp = *p*; *p* += 1; *y* = **temp*;

либо как

y = **p*; *p* += 1;

Выражение может содержать несколько операций с побочным эффектом. Порядок вычислений важен для понимания того, когда проявляется побочный эффект.

Точка последовательных вычислений (*sequence point*) — это точка в программе, где можно точно определить, какие из побочных эффектов уже проявились, а какие — еще нет. Если полное выражение является частью оператора, то точкой, где заведомо выполнились все побочные эффекты его вычисления, — конец этого выражения. Например, в *y* = 37; *x* += *y*; можно быть уверенным, что 37 будет занесено в *y* раньше, чем значение *y* будет извлечено из памяти при вычислении суммы *x* + *y*.

Полное выражение — это выражение инициализации, операторное выражение, выражение в операторе return и управляющее выражение в условном операторе, цикле или операторе *switch* (в том числе каждое из трех выражений оператора *for*).

Кроме того, точки последовательных вычислений могут быть расположены внутри самого выражения:

- при выполнении операции ***x, y*** такая точка находится между вычислением *x* и *y*;
- при выполнении операции ***z ? x : y*** такая точка находится между вычислением *z* и вычислением *x* либо *y*;
- при вызове функции все побочные эффекты вычисления значений ее аргументов проявятся перед выполнением ее тела;
- при выполнении операций ***x && y*** и ***x || y*** такая точка находится между вычислением *x* и вычислением *y*.

Например, в *if* ((*c* = *getchar* ()) != *EOF* && *isprint*(*c*)) вызов функции *isprint* (*c*) произойдет только после того, как переменная *c* получит новое значение.

В стандарте Си между двумя точками последовательных вычислений изменение значения переменной возможно не более одного раза. Например, верно

val = 10 * *val* + (*c* − '0');

но неверно

i = ++*i* + 2;

Другой пример выражения, значение которого стандарт Си считает неопределенным:

++*i**++*i*

Выражение может содержать точки последовательных вычислений, и тем не менее, порядок вычислений не будет однозначным. Например, *f* (*x*) + *g* (*x*) содержит такие точки, однако операция + допускает произвольный порядок вычисления ее операндов.