Основные понятия языка программирования

Структура ЯВУ

[http://kufas.ru]

Алфавит

Алфавит – набор символов, допустимых для записи программ.

Алфавит языка Си включает (5.2.1)

- латинские буквы (A-Z, a-z);
- цифры (0-9);
- 29 графических символов (! " # % & ' ( ) * + , - и т.д.);
- пробел, горизонтальная табуляция, вертикальная табуляция, символ перехода на новую страницу.

Лексемы

Из символов алфавита образуются лексемы. *Лексема (токен)* – это минимальная смысловая единица языка.

К лексемам в языке Си относятся (6.4)

- идентификаторы;
- ключевые слова;
- константы;
- строковые литералы;
- пунктуаторы (punctuators).

Идентификаторы

Идентификатор (имя) – строка символом, используемая для идентификации некоторой сущности в программе.

- В языке Си идентификатор может содержать буквы, цифры и символ подчеркивания.
- Идентификатор не может начинаться с цифры.
- Регистр символов имеет значение.

Правильные имена	**Неправильные имена**
times10 get_next_char _done (!) puts (!)	10times get-next-char while

Ключевые слова

Ключевое слово – это идентификатор, смысл которого зафиксирован правилами языка программирования и который используется для распознавания предложений в программе.

Ключевое слово называется *зарезервированным*, если синтаксис запрещает его использование в качестве идентификатора, определяемого программистом.

Во многих языках содержатся *предопределенные* имена, которые занимают некоторое среднее положение между зарезервированными словами и именами, определяемыми пользователем (см. слайд 5).

Ключевые слова языка Си (c99)

auto	enum	restrict	unsigned
break	extern	return	void
case	float	short	volatile
char	for	signed	while
const	goto	sizeof	_Bool
continue	if	static	_Complex
default	inline	struct	_Imaginary
do	int	switch	
double	long	typedef	
else	register	union	

Константы

Константа (постоянная величина) - это величина, значение которой остается неизменным во время выполнения программы.

Различают целые, вещественные, символьные и строковые константы.

Целые	100, -257, 0xFF, 3L
Вещественные	3.14159, -1e-5, 2.71928f
Символьные	‘a’, ‘b’, ‘\a’
Строковые	“Hello, world\n”

Пунктуаторы

Пунктуатор – символ, который имеет независимое синтаксическое и семантическое значение. В зависимости от контекста он определяет операцию для выполнения.

```
[ ] ( ) { } . ->
++ -- & * + - ~ !
/ % << >> < > <= >= == != ^ | && ||
? : ; ...
= *= /= %= += -= <<= >>= &= ^= |=
, # ##
<: :> <% %> %: %:%:
```

Синтаксис и семантика ЯП

Синтаксис языка программирования – набор правил, описывающий комбинации символов алфавита, считающиеся правильно структурированной программой или её фрагментом. [wikipedia]

Семантика языка программирования – это правила придания смысла синтаксически правильным программам. Эти правила определяют ту последовательность действий вычислительной машины, которую она должна выполнить, работая по данной программе.

Синтаксис ЯП

Для описания синтаксиса языка часто используются формы Бэкуса-Наура.

Пример описания операторов выбора языка Си (6.8.4)

```
selection-statement:
    if ( expression ) statement
    if ( expression ) statement else statement
    switch ( expression ) statement
```

Семантика ЯП

```
for (Initialization Statement; Test Expression; Update Statement)
{
        // Body
}
```

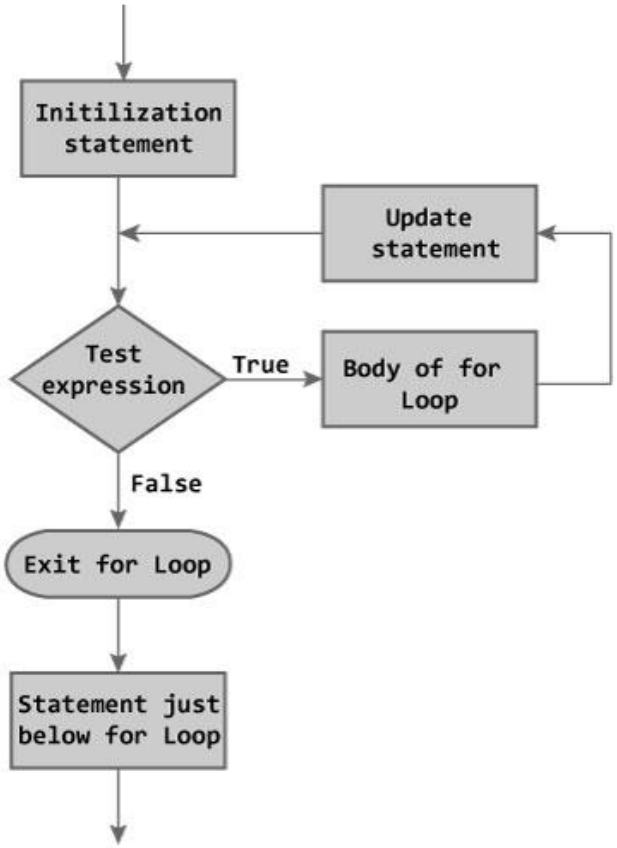


Figure: Flowchart of for Loop

[https://www.programiz.com/c-programming/c-for-loop]

Операции и выражения

Простейшим средством описания действий является *выражение*.

Выражение задает действия, которые вычисляют единственное значение. Состоит из констант, переменных, а также знаков операций и скобок.

Операция - конструкция в языках программирования, аналогичная по записи математическим операциям, то есть специальный способ записи некоторых действий. [wiki]

Элементы данных, к которым применяется операция, называют *операндами*.

Классификация операций

В языке Си операции *по способу записи* подразделяются на

Постфиксные	`<операнд><операция>`	`i++`
Префиксные	`<операция><операнд>`	`&speed`
Инфиксные	`<операнд><операция><операнд>`	`width * length`

по количеству операндов на

Унарные	Один операнд	`-a`
Бинарные	Два операнда	`width * length`
Тернарные	Три операнда	`(a > b) ? a : b`

Побочный эффект

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

Некоторые операции имеют *побочный эффект (side effect)*.

При выполнении этих операций, кроме основного эффекта - вычисления значения - происходят изменения объектов или файлов. Такова, например, операция постфиксного инкремента “i++”.

Приоритет и ассоциативность

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

Правила приоритетов операций при вычислении выражений определяют порядок, в котором выполняются операции, имеющие разные приоритеты.

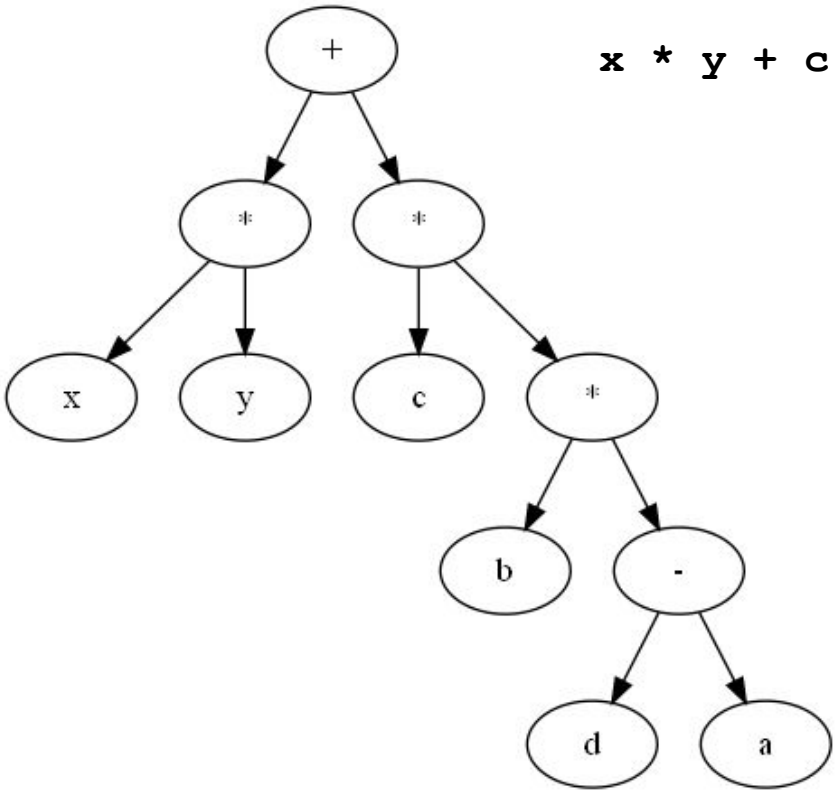
```
a + b * c   =>   a + (b * c)
```

На вопрос о том, какая из операций, имеющих разный приоритет, выполняется первой, отвечают *правила ассоциативности*.

```
a - b + c   =>   ((a - b) + c)   // левоассоциативные
a = b = c   =>   ((a = (b = c))) // правоассоциативные
```

Порядок вычисления подвыражений

В языке Си порядок вычисления подвыражений не определен.




```
x * y + c * b * (d - a)
```

Порядок вычисления подвыражений

Существуют две схемы вычисления *логических выражений*: полная и неполная.

Полная схема означает, что выражения вычисляются полностью слева направо.

По *короткой схеме* вычисление логических выражений прерывается, как только станет известным итоговый результат выражений (истина или ложь).

В языке Си используется короткая схема.

Операторы

Вычисления в программах, написанных на императивных языках программирования, выполняются путем вычисления выражения и присваивания результирующих значений переменным.

Чтобы сделать вычисления гибкими, необходимо наличие (по крайней мере) средств выбора среди альтернативных путей потока управления и средств для организации повторного выполнения набора действий.

Операторы определяют каким образом управление передается из одной части программы другой.

Подпрограммы

Подпрограмма - именованная часть программы, содержащая описание определённого набора действий. Подпрограмма может быть многократно вызвана из разных частей программы.

Функция - это подпрограмма специального вида, которая всегда должна возвращать результат. Вызов функции является, с точки зрения языка программирования, выражением, он может использоваться в других выражениях или в качестве правой части присваивания.

Процедура - это независимая именованная часть программы, которую после однократного описания можно многократно вызвать по имени из других частей программы для выполнения определенных действий.

Преимущества подпрограмм

- Модульность (использование подпрограмм позволяет разделить большую программу на части меньшего размера, которые легче понимать и изменять).
- Закрытость реализации (изменение реализации подпрограммы не влияет на остальную часть программы, если интерфейс подпрограммы не изменился).
- Расширение возможностей языков программирования (создание библиотек).

Литература

1. Орлов С.А. «Теория и практика языков программирования»
2. R.W. Sebesta «Concepts of Programming Languages»
3. Т. Пратт, М. Зелковиц «Языки программирования. Разработка и реализация.»
4. Черновик стандарта C99