

Синтаксис внешних объявлений

Лекция 333

План лекции

- Обзор тела синтаксиса языка Си
- Синтаксис внешних определений на языке Си
- Синтаксис объявлений на языке Си
- Синтаксис инструкций на языке Си

Простые типы данных

- *Тип данных* – это пара, состоящая из множества значений и множества операций над ними
- Языки программирования позволяют строить одни типы данных из других типов данных
- *Простые* типы данных – это типы данных, доступные без построения
- *Составные* типы данных – это типы данных, которые строятся из других типов данных

Простые типы данных языка Си

- Символы
- Целые числа
- Вещественные числа

СИМВОЛЫ

- Тип для хранения кодов символов и 8-битных целых чисел
 - char
 - Стандарт Си не определяет, есть ли знак у значений типа char
 - signed char
 - unsigned char
- Минимальное и максимальное значения CHAR_MIN, CHAR_MAX, UCHAR_MAX в limits.h
- Минимальные стандартные диапазоны
 - signed char -127 ... 127
 - unsigned char 0 ... 255

Целые числа 1/2

- C89
 - [signed|unsigned] [short|long] int
- Дополнительно в C99 и новее
 - [signed|unsigned] long long [int]
- Знаковые типы: [signed] short int, [signed] int, [signed] long int, [signed] long long int
- Некоторые компиляторы поддерживают 128-битные целые числа
 - Нет в стандарте Си

Целые числа 2/2

Варианты имени	Диапазон значений в limits.h	Минимальный стандартный диапазон
[signed] short [int]	SHRT_MIN ... SHRT_MAX	-32767 ... 32767
unsigned short [int]	0 ... USHRT_MAX	0 ... 65535
int signed [int]	INT_MIN ... INT_MAX	как у short
unsigned [int]	0 ... UINT_MAX	как у unsigned short
[signed] long [int]	LONG_MIN ... LONG_MAX	$-(2^{31}-1) \dots 2^{31}-1$
unsigned long [int]	0 ... ULONG_MAX	$0 \dots 2^{32}-1$
[signed] long long [int]	LLONG_MIN ... LLONG_MAX	$-(2^{63}-1) \dots 2^{63}-1$ (только C99 и новее)
unsigned long long [int]	0 ... ULLONG_MAX	$0 \dots 2^{64}-1$ (только C99 и новее)

sizeof(char)	==	sizeof(unsigned char) <=
<= sizeof(short)	==	sizeof(unsigned short) <=
<= sizeof(int)	==	sizeof(unsigned) <=
<= sizeof(long)	==	sizeof(unsigned long) <=
<= sizeof(long long)	==	sizeof(unsigned long long)

Вещественные числа

- Типы для хранения вещественных чисел
 - float
 - double
 - long double
- `sizeof(float) <= sizeof(double) <= sizeof(long double)`
- Границы диапазонов `FLT_MIN`, `FLT_MAX` и т.п. в файле `float.h`
- Минимальный стандартный диапазон $-10^{37} \dots 10^{37}$

Машинное представление значений простых типов

- Символы
- Целые числа
- Вещественные числа

МП char, signed char, unsigned char 1/3

- 1 байт памяти,
 - signed char целые числа от -128 до 127
 - unsigned char целые числа от 0 до 255
- Значения типов char, signed char, unsigned char кодируют СИМВОЛЫ
- Соответствие значений и символов определяется кодировкой ОС

МП char, signed char, unsigned char 2/3

- Кодировка CP866 (MS DOS)

	00	10	20	30	40	50	60	70
0		»		0	@	P	`	p
1	☺	◀	!	1	A	Q	a	q
2	☹	↑	"	2	B	R	b	r
3	♥	!!	#	3	C	S	c	s
4	♦	¶	\$	4	D	T	d	t
5	♣	§	%	5	E	U	e	u
6	♠	=	&	6	F	V	f	v
7	•	±	'	7	G	W	g	w
8	☐	↑	<	8	H	X	h	x
9	○	↓	)	9	I	Y	i	y
A	◻	→	*	:	J	Z	j	z
B	♂	←	+	;	K	[	k	{
C	♀	└	,	<	L	\	l	!
D	℞	↔	-	=	M	]	m	}
E	☾	▲	.	>	N	^	n	~
F	※	▼	/	?	O	_	o	△

	80	90	A0	B0	C0	D0	E0	F0
0	A	P	a	☐	└	└	P	≡
1	Б	С	б	▀	└	└	С	±
2	В	Т	в	▄	└	└	Т	>
3	Г	У	г		└	└	У	<
4	Д	Ф	д	└	-	└	Ф	└
5	Е	Х	е	└	└	└	Х	J
6	Ж	Ц	ж	└	└	└	Ц	÷
7	З	Ч	з	└	└	└	Ч	≈
8	И	Ш	и	└	└	└	Ш	°
9	Й	Щ	й	└	└	└	Щ	.
A	К	Ь	к		└	└	Ь	.
B	Л	Ы	л	└	└	■	Ы	√
C	М	Ъ	м	└	└	■	Ъ	n
D	Н	Э	н	└	=	■	Э	2
E	О	Ю	о	└	└	■	Ю	●
F	П	Я	п	└	└	■	Я	

МП char, signed char, unsigned char 3/3

- Linux (KOI8)

Если в тексте в KOI-8 убирать восьмой бит каждого символа, то транслит с антикапсом.

Например, Русский Текст --> rUSSKIj tEkSt.

Code	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
80	:															
90	:															
A0	:			Ё												
B0	:			ё												
C0	:	ю	а	б	в	г	д	е	ж	з	и	й	к	л	м	н
D0	:	п	я	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь
E0	:	Ю	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н
F0	:	П	Я	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь

- Win 1251

Code	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
80	:															
90	:															
A0	:								Ё							
B0	:								ё							
C0	:	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О
D0	:	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э
E0	:	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о
F0	:	п	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э

- Mac OS

Code	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
80	:	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О
90	:	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э
A0	:															
B0	:															
C0	:															
D0	:															
E0	:	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о
F0	:	п	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э

МП целых чисел без знака

- *Двоичная запись* числа $Ч$ – это последовательность битов $b_n \dots b_1 b_0$ такая, что

$$Ч = 2^n \cdot b_n + \dots + 2^1 \cdot b_1 + 2^0 \cdot b_0$$

- МП целого числа x без знака – это *двоичная запись* числа $x \bmod 2^{8 \cdot \text{sizeof}(x)}$

МП целых чисел со знаком 1/2

- МП числа x со знаком
 - двоичная запись $x \bmod 2^{8 \cdot \text{sizeof}(T)}$, если $x \geq 0$
 - *дополнительный код* $|x|$ -- двоичная запись $2^{8 \cdot \text{sizeof}(T)} - |x| \bmod 2^{8 \cdot \text{sizeof}(T)}$, если $x < 0$
- Докажите, что $\text{МП}(\text{МП}(x) + \text{МП}(y)) = \text{МП}(x + y)$, $\text{МП}(\text{МП}(x) * \text{МП}(y)) = \text{МП}(x * y)$

МП целых чисел со знаком 2/2

Построение дополнительного кода

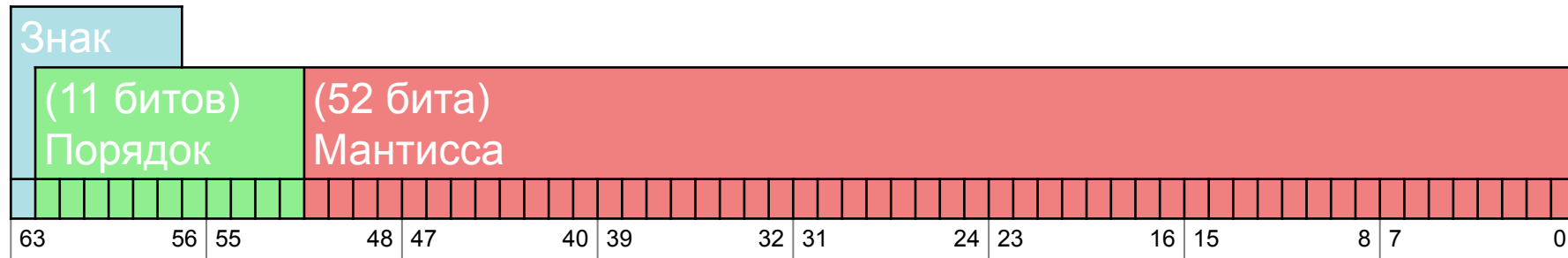
- Вход $b[n]$ – двоичная запись $|x|$
- Выход $d[n]$ – дополнительный код $|x|$
- Алгоритм

```
for (i = 0; i < n; i = i+1)
    d[i] = 1-b[i];
for (i = 0; i < n && d[i] == 1; i = i+1)
    d[i] = 0;
if (i < n) d[i] = 1;
```

МП вещественных чисел 1/3

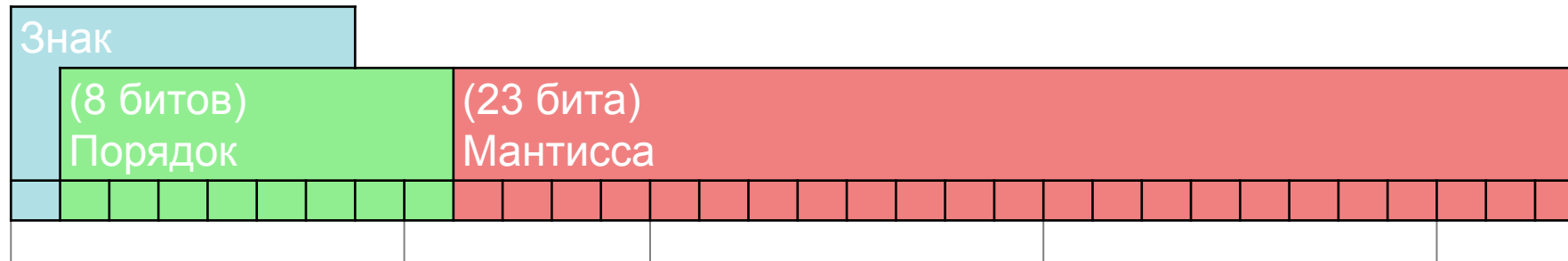
- Числа с плавающей точкой – это числа вида $S \cdot M \cdot 2^P$
- S – знак +1 или -1
- M – мантисса, $x/2^{mb}$ от 0 до 1
 - mb – число битов в мантиссе
 - x – целое от 0 до $2^{mb}-1$
- P – порядок

МП double – стандарт IEEE 754



Порядок	Мантисса 0	Мантисса != 0	Формула
0x000	0 и -0	Денормализов. числа	$(-1)^{\text{знак}} \cdot 2^{\text{порядок}-1022} \cdot (0.\text{мантисса})_{(2)}$
0x001 ... 0x7fe	Нормализованные числа		$(-1)^{\text{знак}} \cdot 2^{\text{порядок}-1023} \cdot (1.\text{мантисса})_{(2)}$
0x7ff	$+\infty$ или $-\infty$	NaN	
$3\text{ff}0\ 0000\ 0000\ 0000_{(16)} = 1$			
$0000\ 0000\ 0000\ 0000_{(16)} = 0$			
$7\text{ff}0\ 0000\ 0000\ 0000_{(16)} = \infty$			
$3\text{ff}0\ 0000\ 0000\ 0001_{(16)} \approx 1.00000000000000002$			
$8000\ 0000\ 0000\ 0000_{(16)} = -0$			
$\text{ff}f0\ 0000\ 0000\ 0000_{(16)} = -\infty$			
$3\text{fd}5\ 5555\ 5555\ 5555_{(16)} \approx 1/3$			

МП float – стандарт IEEE 754



Порядок	Мантисса 0	Мантисса != 0	Формула
0x00	0 и -0	Денормализов. числа	$(-1)^{\text{знак}} \cdot 2^{\text{порядок}-126} \cdot (0.\text{мантисса})_{(2)}$
0x01 ... 0xfe	Нормализованные числа		$(-1)^{\text{знак}} \cdot 2^{\text{порядок}-127} \cdot (1.\text{мантисса})_{(2)}$
0xff	$+\infty$ или $-\infty$	NaN	

МП данных простых типов -- разное

- Значение переменной простого типа В хранится по адресу *выровненному на* (кратному) `sizeof(B)` байтов

Грамматика языка Си



Единица трансляции 1/1

translation-unit:

external-declaration

translation-unit **external-declaration**

external-declaration:

function-definition

declaration

function-definition:

declaration-specifiers **declarator** **declaration-list** opt **compound-statement**

declaration-list:

declaration

declaration-list **declaration**

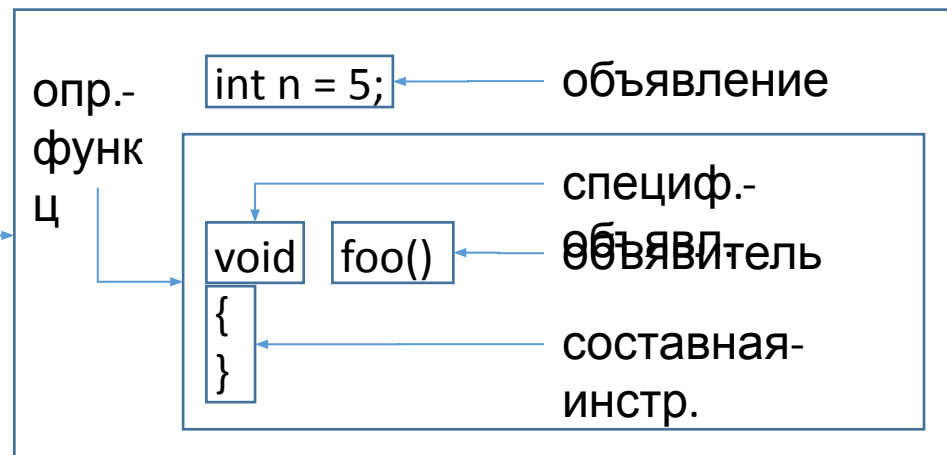
единица-
трансл.

Здесь

- единица-трансляции
- внешнее-объявление
- определение-функции
- список-объявлений

Далее

- объявление
- спецификаторы-
объявления
- объявитель
- составная-инструкция



Объявление 1/3

declaration:

declaration-specifiers **init-declarator-list** opt ;

declaration-specifiers:

storage-class-specifier **declaration-specifiers** opt

type-specifier **declaration-specifiers** opt

type-qualifier **declaration-specifiers** opt

function-specifier **declaration-specifiers** opt

Здесь

- объявление
- спецификаторы-объявления

Далее

- список-объявителей-инициализаторов
- спецификатор-класса-памяти
- спецификатор-типа
- квалификатор-типа
- спецификатор-функции
- объявитель
- составная-инструкция

Объявление 2/3

init-declarator-list:

init-declarator

init-declarator-list , **init-declarator**

init-declarator:

declarator

declarator = **initializer**

initializer:

assignment-expression

{ **initializer-list** }

{ **initializer-list** , }

initializer-list:

designation opt **initializer**

initializer-list , **designation** opt **initializer**

Здесь

- список-объявителей-инициализаторов
- объявитель-инициализатор
- инициализатор
- список-инициализаторов

Далее

- обозначитель
- спецификатор-класса-памяти
- спецификатор-типа
- квалификатор-типа
- спецификатор-функции
- объявитель
- выражение-присваивание
- составная-инструкция

Объявление 3/3

designation:

designator-list =

designator-list:

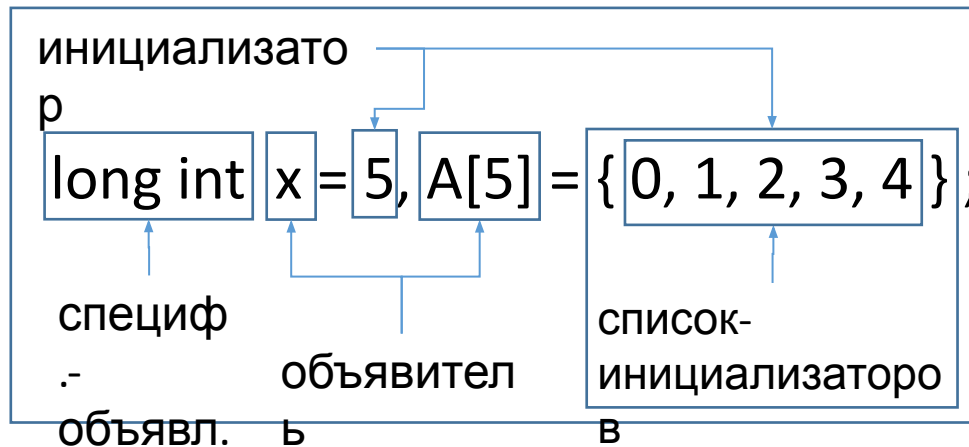
designator

designator-list **designator**

designator:

[**constant-expression**]

. **identifier**



Здесь

- обозначитель
- список-обозначителей

Далее

- спецификатор-класса-памяти
- спецификатор-типа
- квалификатор-типа
- спецификатор-функции
- объявитель
- константное-выражение
- выражение-присваивание
- составная-инструкция

объявлени
е

Спецификаторы и квалификатор

storage-class-specifier:		type-qualifier:	
typedef		const	
extern		restrict	
static		volatile	
auto		function-specifier:	
register		inline	
type-specifier:	void		
	char		
	short		
	int		
	long		
	float		
	double		
	signed		
	unsigned		
	_Bool		
	_Complex		

struct-or-union-specifier

enum-specifier

typedef-name

Здесь

- спецификатор-класса-памяти
- спецификатор-типа
- квалификатор-типа
- спецификатор-функции

Далее

- объявитель
- константное-выражение
- спецификатор-struct-или-union
- спецификатор-enum
- имя-typedef
- выражение-присваивание
- составная-инструкция

Объявитель 1/3

declarator:

pointer opt **direct-declarator**

direct-declarator:

identifier

(**declarator**)

x-team.ru) **direct-declarator** [**type-qualifier-list** opt **assignment-expression**] основная инструкция

direct-declarator [static **type-qualifier-list** opt **assignment-expression**]

direct-declarator [**type-qualifier-list** static **assignment-expression**]

direct-declarator [**type-qualifier-list** opt *]

direct-declarator (**parameter-type-list**)

direct-declarator (**identifier-list** opt)

Здесь

- объявитель
- непосредственный-объявитель

Далее

- указатель
- список-типов-параметров
- список-идентификаторов
- список-квалификаторов-типа
- константное-выражение
- спецификатор-struct-или-union
- спецификатор-enum
- имя-typedef
- выражение-присваивание

Объявление идентификаторов, имеющих составной тип (массив, функция, указатель, struct, union, enum)

Объявитель 2/3

pointer:

* **type-qualifier-list** opt

* **type-qualifier-list** opt **pointer**

type-qualifier-list:

type-qualifier

type-qualifier-list **type-qualifier**

parameter-type-list:

parameter-list

parameter-list , ...

identifier-list:

identifier

identifier-list , **identifier**

parameter-list:

parameter-declaration

parameter-list , **parameter-declaration**

parameter-declaration:

declaration-specifiers **declarator**

declaration-specifiers **abstract-declarator** opt

Здесь

- указатель
- список-типов-параметров
- список-квалификаторов-типа
- список-параметров
- объявление-параметра
- список-идентификаторов

Далее

- абстрактный-объявитель
- константное-выражение
- спецификатор-struct-или-union
- спецификатор-enum
- имя-typedef
- выражение-присваивание
- составная-инструкция

Объявитель 3/3

abstract-declarator:

pointer

pointer opt **direct-abstract-declarator**

direct-abstract-declarator:

(**abstract-declarator**)

direct-abstract-declarator opt [**type-qualifier-list** opt **assignment-expression** opt]

direct-abstract-declarator opt [static **type-qualifier-list** opt **assignment-expression**]

direct-abstract-declarator opt [**type-qualifier-list** static **assignment-expression**]

direct-abstract-declarator opt [*]

direct-abstract-declarator opt (**parameter-type-list** opt)

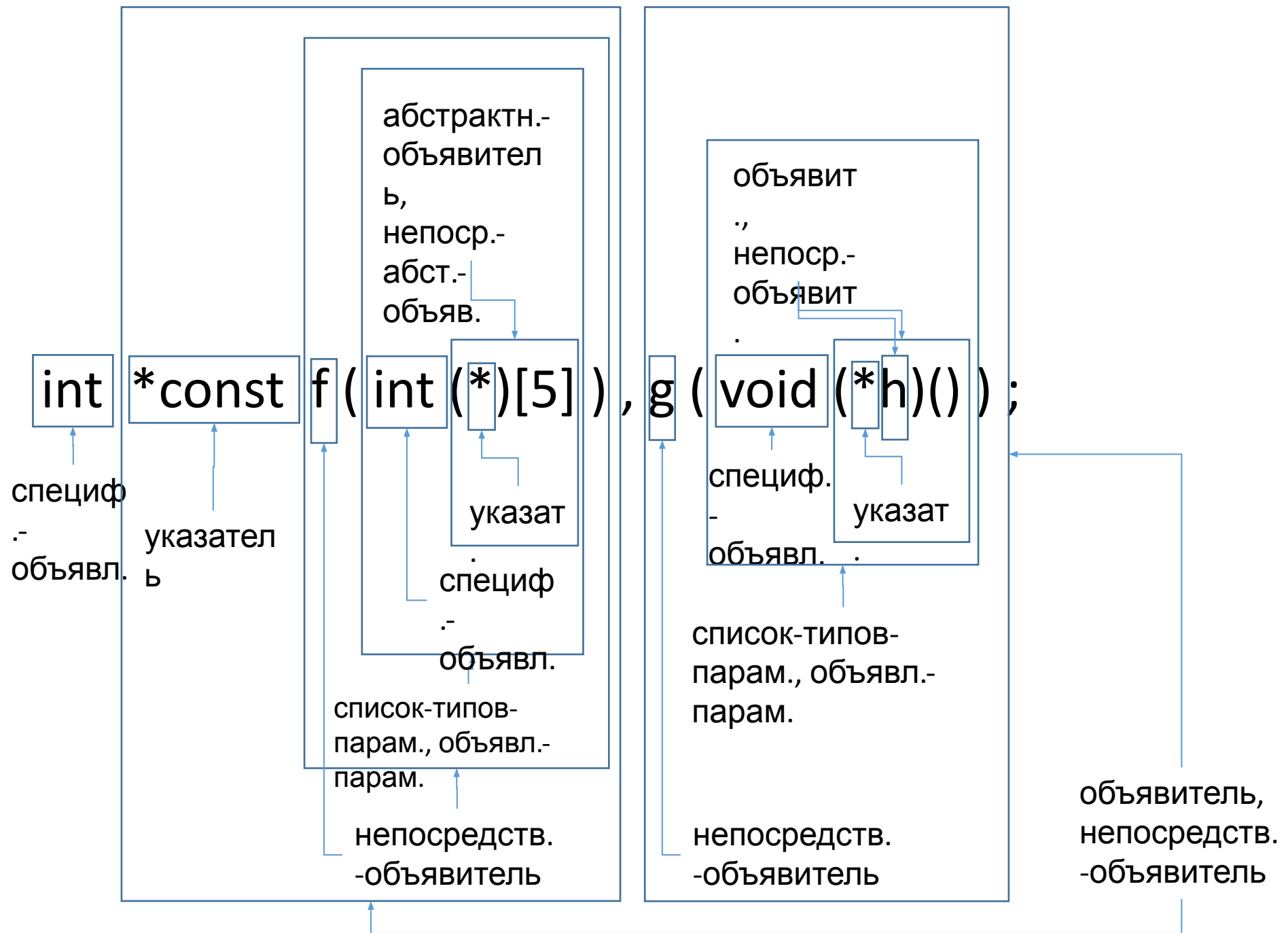
Здесь

- абстрактный-объявитель
- непосредственный-
абстрактный-объявитель

Далее

- константное-выражение
- спецификатор-struct-
или-union
- спецификатор-enum
- имя-typedef
- выражение-присваивание
- составная-инструкция

Объявление типа параметра
функции без указания имени
параметра



enum

enum-specifier:

enum **identifier** opt { **enumerator-list** }

enum **identifier** opt { **enumerator-list** , }

enum **identifier**

enumerator-list:

enumerator

enumerator-list , **enumerator**

enumerator:

enumeration-constant

enumeration-constant = **constant-expression**

Здесь

- спецификатор-enum
- имя-typedef – a.k.a. идентификатор

Далее

- константное-выражение
- спецификатор-struct-или-union
- выражение-присваивание
- составная-инструкция

- Элементы enum являются константами и имеют тип int

- Значения типа enum могут быть преобразованы в char или один из целых типов (со знаком или без знака)

- Этот тип должен быть достаточен для хранения значений всех элементов enum

- Конкретный тип выбирается

struct, union 1/2

Здесь

- спецификатор-struct-или-union
- struct-или-union
- список-описания-struct
- описание-struct

struct-or-union-specifier:

struct-or-union **identifier** opt { **struct-declaration-list** }

struct-or-union **identifier**

struct-or-union:

struct

union

struct-declaration-list:

struct-declaration

struct-declaration-list **struct-declaration**

struct-declaration:

specifier-qualifier-list **struct-declarator-list** ;

Далее

- список-спецификаторов-квалификаторов
- список-описателя-struct
- константное-выражение
- выражение-присваивание
- составная-инструкция

struct, union 2/2

struct-declarator-list:

struct-declarator

struct-declarator-list , **struct-declarator**

struct-declarator:

declarator

declarator opt : **constant-expression**

```
struct Small_3D_Point {  
    unsigned int x:10, y:10, z:10;  
};
```

Здесь

- список-описателя-struct
- список-спецификаторов-квалификаторов

Далее

- константное-выражение
- выражение-присваивание
- составная-инструкция

- Через : задается ширина битового поля внутри значений типа _Bool (C99), signed int, unsigned int или типа, определенного реализацией компилятора
- Ширина битового поля должна быть ≥ 0 и не превышать числа битов в значении, внутри которого находится битовое поле
- Значением битового поля является целое число со знаком или без знака
- Битовые поля последовательно упаковываются в
- Если ширина поля равна нулю, то объявитель должен отсутствовать;
такое поле — пустое поле внутри

Составная инструкции

compound-statement:

{ **block-item-list** opt }

block-item-list:

block-item

block-item-list **block-item**

block-item:

declaration

statement

Здесь

- составная-инструкция
- список-блоков
- блок
- инструкция

Далее

- выражение-присваивание
- константное-выражение
- помеченная-инструкция
- выражение-инструкция
- инструкция-выбора
- инструкция-повторения
- инструкция-перехода

statement:

labeled-statement

compound-statement

expression-statement

selection-statement

iteration-statement

jump-statement

Заключение

- Простые типы данных
 - Ограничения на простые типы данных
- Машинное представление простых типов данных
- Обзор тела синтаксиса языка Си
- Синтаксис внешних определений на языке Си
- Синтаксис объявлений на языке Си
- Синтаксис инструкций на языке Си