Сложные объявления

Чтение сложных объявлений

Не возникает проблем с чтением следующих объявлений:

```
int foo[5];
char *foo;
double foo(void).
```

Но как только объявление становятся сложнее, трудно точно сказать что это. Например,

```
char *(*(**foo[][8])())[];
```

Чтение сложных объявлений
(замена элементов объявления фразами)

[]	массив типа …
[N]	массив из N элементов типа…
(type)	функция, принимающая аргумент типа type и возвращающая …
*	указатель на …

Чтение сложных объявлений
(правила)

- «Декодирование» объявления выполняется «изнутри наружу». При этом отправной точкой является идентификатор.
- Когда сталкиваетесь с выбором, отдавайте предпочтение «[]» и «()», а не «*», т.е.

  `*name[]` — «массив типа», не «указатель на»

  `*name()` — «функция, принимающая», не «указатель на»

  При этом «()» могут использоваться для изменения приоритета.

Чтение сложных объявлений
(примеры)

1. `int *(*x[10])(void);`

2. `char *(*(**foo[][8])())[];`

3. `void (*signal(int, void (*fp)(int)))(int);`

Чтение сложных объявлений

(семантические ограничения)

- Невозможно создать массив функций.

```
int a[10](int);
```

- Функция не может возвращать функцию.

```
int g(int)(int);
```

- Функция не может вернуть массив.

```
int f(int)[];
```

- В массива только левая лексема [] может быть пустой.
- Тип void ограниченный.

```
void x;        // ошибка
void x[5];     // ошибка
```

Чтение сложных объявлений
(использование typedef для упрощения)

```
int *(*x[10])(void);

typedef int* func_t(void);
typedef func_t* func_ptr;
typedefe func_ptr* funt_ptr_arr[10];

funt_ptr_arr x;
```