

ИНТЕРПРЕТАЦИЯ СЛОЖНЫХ ОПИСАНИЙ (новое)

Язык Си позволяет вам создавать **сложные описания** данных. При создании описания мы используем имя(идентификатор), и модификаторы(имени):

Модификатор (имени)	слева или справа от имени	Значение(смысл) имени
*	слева	Указатель
()	справа	Функция
[]	справа	Массив

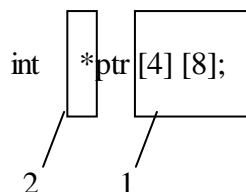
Язык Си позволяет использовать **одновременно более одного модификатора**:

- 1) `int ptr[4][8];` // массив 4x8 элементов (2-мерный) значений типа `int`
- 2) `int **ptr;` // указатель на указатель на значение типа `int`
- 3) `int *ptr[8];` // 8-элементный массив указателей на значение типа `int`
- 4) `int (*ptr)[8];` // указатель на 8-элементный массив элементов типа `int`
- 5) `int *ptr[4][8];` // 4-элементный массив из 8-элементных массивов указателей на значение типа `int`
- 6) `int (*ptr)[4][8];` // указатель на массив 4x8 элементов типа `int`

Для распутывания этих описаний надо знать, **в каком порядке** следует применять модификаторы. Следующие **правила** говорят об этом:

- 1) Модификаторы `[]` и `()` имеют приоритет выше, чем `*`. (т.е. начинать анализ описания надо **справа от имени**)
- 2) Модификаторы `[]` и `()` имеют одинаковый приоритет и рассматриваются (друг относительно друга) слева направо.
- 3) К интерпретации `*` надо переходить лишь тогда, когда закончатся все `()` и `[]` справа. Это означает, что анализ конструкций языка Си должен производиться в целом в направлении справа налево, т.е. сначала все справа от имени, а потом то, что слева (если нет скобок).
- 4) Круглые скобки (кроме указания на то, что имя является функцией) используются (должны использоваться) для объединения частей описания, которым надо назначить самый высокий приоритет.
- 5) Чем ближе модификатор стоит к идентификатору, тем выше его приоритет. Это означает, что анализ надо начинать с модификатора, **наиболее близкого к идентификатору** (а из тех, что ближе, выбирать вначале тот, который справа) .

Рассмотрим применение этих правил на следующем примере:



В этом примере:

- то, что объединено в прямоугольник 1, рассматривается в 1-ю очередь (см. правило №1).
- из того, что в прямоугольнике 1, сначала рассматривается `[4]`, потом `[8]` (правило 2).
- то, что объединено в прямоугольник 2, надо начинать просматривать после того, как будет просмотрено все, что в прямоугольнике 1 (правило 3).

Т.о. согласно приведенным правилам имя ptr является 4-элементным массивом (левый модификатор [4]) из 8-элементных массивов (правый модификатор [8])указателей (модификатор *) на значение типа int.

Другой пример - в описании

```
int (*ptr)[4][8];
```

круглые скобки говорят о том , что модификатор * должен иметь первый приоритет, а это означает, что ptr является указателем (на массив 4x8 значений типа int).

Рассмотренные правила позволяют интерпретировать и следующие описания:

7) char *f(); // функция, возвращающая указатель на значение типа char

8) char (*f) (); // указатель на функцию, возвращающую значение типа char

9) char *f() [4]; // функция, возвращающая 4-элементный массив указателей на значение типа char

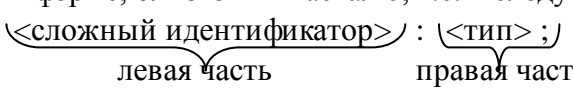
10) char *f[4] (); // 4-элементный массив функций, возвращающих указатель на значение типа char .

11) char (*f[4])(); // массив из 4-х указателей на функцию, возвращающую значение типа char/

12) int *(*ptr[4])[8]; // массив из 4-х указателей на 8-элементный массив из указателей на значение типа int.

13) int *(*(*x[4])())[8]; // массив из 4-х указателей на функцию, возвращающую указатель на 8-элементный массив из указателей на значение типа int.

Приведенный метод интерпретации сложных описаний является неформальным (эмпирическим) и может вызвать затруднения при увеличении сложности описаний. В случае затруднений с применением описанного выше способа интерпретации целесообразно применять **второй способ** интерпретации, основанный на:

- переходе от представления описания данных в форме, близкой к Си, т.е. вида
<тип> <сложный идентификатор>
к форме, близкой к Паскалю, т.е. в следующем виде

левая часть правая часть
- поэтапном понижении сложности описания в левой части за счет последовательного переноса модификаторов из левой в правую часть с одновременной заменой символа * на слово «указатель», () на слово «функция», а [] – на слово «массив».

При интерпретации сложных описаний этим методом необходимо формально применять следующие **правила**:

1). Тип того, что самое левое в сложном описании на Си, переносится в правую часть первым, т.е. конструкция в стиле Си вида

<тип> <остаток описания>

заменяется на конструкцию в стиле Паскаля вида

<остаток описания> : <тип>

2). Символы описания (модификаторы) *, () и [] переносятся из левой части в правую (добавляются к правой части слева), так чтобы вместо символов *, () и [] в левой части появились соответствующие слова(ссылка, функция и массив) в правой части

3). Если в описании идентификатор окружают слева и справа символы *. [] и (), то в правую часть в первую очередь должен переноситься символ *.

- 4) Если слева в описании символов * больше нет (не осталось), а справа от имени имеются () и [], то они переносятся в правую часть последовательно справа налево.
- 5). Если часть описания заключена в скобки, то эти скобки не раскрываются до тех пор, пока не будут перенесены в правую часть все символы *, () и [] справа и слева от этих скобок. Если же скобки вложенные, то вначале раскрываются самые внешние скобки, потом – более внутренние и т.д.

Рассмотрим соответствующие примеры применения этих правил по мере увеличения сложности описаний:

- 1). `int x` - $\implies x : \text{int}$
- 2). `int *x` $\implies *x : \text{int} \implies x : \text{указатель на значение типа int}$
- 3). `int x[]` $\implies x[] : \text{int} \implies x : \text{массив из значений типа int}$
- 4). `int x()` $\implies x() : \text{int} \implies x : \text{функция, возвращающая значение типа int}$
- 5). `int *x[]` $\implies *x[] : \text{int} \implies x[] : \text{указатель на значение типа int} \implies x : \text{массив указателей на значение типа int}$
- 6). `int (*x)[]` $\implies (*x)[] : \text{int} \implies (*x) : \text{массив из значений типа int} \implies x : \text{указатель на массив значений типа int}$
- 7). `int *x()` $\implies *x() : \text{int} \implies x() : \text{указатель на значение типа int} \implies x : \text{функция, возвращающая указатель на значение типа int}$
- 8). `int (*x)()` $\implies (*x)() : \text{int} \implies (*x) : \text{функция, возвращающая значение типа int} \implies x : \text{указатель на функцию, возвращающую значение типа int}$
- 9). `int x[]()` $\implies x[]() : \text{int} \implies x[] : \text{функция, возвращающая значение типа int} \implies x : \text{массив функций, возвращающих значение типа int}$
- 10) `int (*x[])()` $\implies (*x[])() : \text{int} \implies (*x[]) : \text{функция, возвращающая значение типа int} \implies x[] : \text{указатель на функцию, возвращающую значение типа int} \implies x : \text{массив указателей на функцию, возвращающую значение типа int}$
- 11) `int ((*x()) []) ()` $\implies ((*x()) []) () : \text{int} \implies ((*x()) []) : \text{функция, возвращающая значение типа int} \implies (*x()) [] : \text{указатель на функцию, возвращающую значение типа int} \implies (*x()) : \text{массив указателей на функцию, возвращающую значение типа int} \implies x() : \text{указатель на массив указателей на функцию, возвращающую значение типа int} \implies x : \text{функция, возвращающая указатель на массив указателей на функцию, возвращающую значение типа int}$

NB: можно еще рассмотреть (дома для закрепления сказанного):

```
int *x[4]()
int (*x[4])()
int *(*x[4])()
int **(*x[4])()()
int *(*(*x[4])())[8]
```

Похожие на первый из двух способов интерпретации правила из книги Романовской по программированию на языке Си для ПЭВМ ЕС звучат так:

- 1) если есть [] и (), то посмотреть их слева направо;
- 2) если есть *, то рассмотреть
- 3) если встретилась закрывающая круглая скобка, то вначале надо применить все правила внутри круглых скобок, а потом идти далее.

Мы рассмотрели задачу **анализа** сложных описаний на языке Си. Существует и обратная ей задача **синтеза** таких описаний на языке Си на базе словесных описаний.

Пример №1:

Пусть, надо получить на языке Си описание для конструкции, описываемой следующими словами: «**x** есть **функция, возвращающая указатель на массив указателей на функцию, возвращающую значение типа char**». Попробуем синтезировать описание на Си поэтапно, также как в задаче анализа, только перенося описания из правой части в левую и заменяя слова указатель, массив и функция на символы *, { } и () соответственно):

```
<== x : функция, возвращающая указатель на массив указателей на функцию, возвращающую
      значение типа char
<== x( ) : указатель на массив указателей на функцию, возвращающую значение типа char
<== * x( ) : массив указателей на функцию, возвращающую значение типа char
<== (*x( )) [ ] : указатель на функцию, возвращающую значение типа char
<== *(*x( )) [ ] : функция, возвращающая значение типа char
<== *(*x( )) [ ] ( ) : char
<== char (*(*x( )) [ ] ( ) .
```

Пример №2:

Пусть ставится задача сконструировать описание для следующего случая:
массив указателей на функцию, возвращающую указатель на массив из значений типа int.

Решение:

Идем по словесному описанию слева направо и ставим справа от имени () и [], а слева - *.
Если надо, то группируем нужные описания с помощью круглых скобок.
Последним записывается тип (слева).
В итоге получим: **int ((*имя[])())[]**

Для упрощения анализа и синтеза сложных описаний на языке Си имеется сайт **cdecl.org**, на котором по-английски, естественно, приводятся интерпретации описаний на Си. Так на описание
char ((*x())[5])()

выводится ответ

declare x as function returning pointer to array 5 of pointer to function returning char

А на запрос

char ((*x[3])())[5]

выводится ответ

declare x as array 3 of pointer to function returning pointer to array 5 of char.