#### 本课件为重点



#### 对于复合类型的声明语句时,先记录类型表达式

- > 基本类型是类型表达式
  - > integer
  - > real
  - > char
  - > boolean
  - ➤ type\_error (出错类型)
  - ► void (无类型)

- > 基本类型是类型表达式
- >可以为类型表达式命名,类型名也是类型表达式
- 》将类型构造符(type constructor)作用于类型表达式可以构成新的类型表达式
  - > 数组构造符array
    - $\triangleright$  若T是类型表达式,则array(I,T)是类型表达式(I是一个整数)

类型	类型表达式
<i>int</i> [3]	array (3, int)
int [2][3]	<pre>array (2, array(3,int) )</pre>

- >基本类型是类型表达式
- >可以为类型表达式命名,类型名也是类型表达式
- ▶将类型构造符(type constructor)作用于类型表达式可以构成新的类型表达式
  - > 数组构造符array
  - >指针构造符pointer
    - $\triangleright$  若T 是类型表达式,则 pointer(T)是类型表达式,它表示一个指针类型

- >基本类型是类型表达式
- >可以为类型表达式命名,类型名也是类型表达式
- 》将类型构造符(type constructor)作用于类型表达式可以构成新的类型表达式
  - > 数组构造符array
  - >指针构造符pointer
  - ➤笛卡尔乘积构造符×
    - $\triangleright$  若 $T_1$  和 $T_2$ 是类型表达式,则笛卡尔乘积 $T_1 \times T_2$  是类型表达式

- > 基本类型是类型表达式
- >可以为类型表达式命名,类型名也是类型表达式
- ▶将类型构造符(type constructor)作用于类型表达式可以构成新的类型表达式
  - > 数组构造符array
  - >指针构造符pointer
  - ➤笛卡尔乘积构造符×
  - ▶函数构造符→
    - > 若 $T_1$ 、 $T_2$ 、...、 $T_n$ 和R是类型表达式,则 $T_1 \times T_2 \times ... \times T_n \to R$ 是类型表达式

- >基本类型是类型表达式
- >可以为类型表达式命名,类型名也是类型表达式
- 》将类型构造符(type constructor)作用于类型表达式可以构成新的类型表达式
  - > 数组构造符array
  - >指针构造符pointer
  - ➤笛卡尔乘积构造符×
  - ▶函数构造符→
  - ▶记录构造符record
    - $\triangleright$  若有标识符 $N_1$ 、 $N_2$ 、...、 $N_n$ 与类型表达式 $T_1$ 、 $T_2$ 、...、 $T_n$ ,则  $record((N_1 \times T_1) \times (N_2 \times T_2) \times ... \times (N_n \times T_n))$  是一个类型表达式

# 例

▶设有C程序片段:

```
struct stype
{ char[8] name;
 int score;
};
stype[50] table;
stype* p;
```

- ▶和stype绑定的类型表达式
  - $\rightarrow$  record ( (name×array(8, char)) × (score × integer) )
- > 和table绑定的类型表达式
  - > *array* (50, *stype*)
- ▶和p绑定的类型表达式
  - > pointer (stype)





#### 局部变量的存储分配

- 》对于声明语句,语义分析的主要任务就是收集标识符的类型等属性信息,并为每一个名字分配一个相对地址
  - 从类型表达式可以知道该类型在运行时刻所需的存储单元数量 称为类型的宽度(width)
  - ▶ 在编译时刻,可以使用类型的宽度为每一个名字分配一个相对 地址
- > 名字的类型和相对地址信息保存在相应的符号表记录中

变量声明语句的SDT enter(name, type, offset):在符号表中为 名字name创建记录,将name的类型设置 为type,相对地址设置为offset

2 
$$D \rightarrow T \text{ id}; \{ \text{ enter}(\text{ id.lexeme}, \text{T.type}, \text{ offset}); \\ \text{offset} = \text{offset} + \text{T.width}; \} D$$

4 
$$T \rightarrow B$$
 {  $t = B.type$ ;  $w = B.width$ ;}  
 $C$  {  $T.type = C.type$ ;  $T.width = C.width$ ; }

	B.type = int;	B.width = 4	;}假设整型	变量占4	个字节
--	---------------	-------------	--------	------	-----

$$\bigcirc B \rightarrow \text{real}\{B.type = real; B.width = 8;\}$$

$$⊗$$
  $C → ε$  {  $C.type=t$ ;  $C.width=w$ ; }

<b>9</b> $C$ → [num] $C_1$ { $C.type = array(num.val,$	$C_1$ .type);
$C.width = num.val * C_1.width; $	

符号	综合属性
В	type, width
<i>C</i>	type, width
T	type, width

变量	作用
offset	下一个可用的相对地址
t, w	将类型和宽度信息从语法分析树中的 $B$ 结点传递到对应于产生式 $C  ightarrow \epsilon$ 的结点

#### 判断是否为LL1:相同左部可选集是否相同

# 例: "real x; int i;"的语法制导翻译

```
\bigcirc P \rightarrow \{ offset = 0 \} D
2D \rightarrow T \text{ id}; \{ enter( id.lexeme, T.type, offset ); \}
          offset = offset + T.width; D
                                                                                                                        enter(x, real, 0)
\mathfrak{D} \rightarrow \varepsilon
(4)T \rightarrow B \quad \{ t = B.type; w = B.width; \}
                                                                                                                              enter(i, int, 8)
          C \{ T.type = C.type; T.width = C.width; \}
                                                                                                          \{a\} D
\textcircled{5}T \rightarrow \uparrow T_1 \{ T.type = pointer(T_1.type); T.width = 4; \}
\textcircled{6}B → int { B.type = int; B.width = 4; }
                                                                                     T type=real
\bigcirc B \rightarrow \text{real} \{ B. type = real; B. width = 8; \}
                                                                                                                       {a}
                                                                                         width=8
\otimes C \rightarrow \varepsilon \{ C.type=t; C.width=w; \}
\mathfrak{G} C \to [\operatorname{num}] C_1 \{ C.type = \operatorname{array}(\operatorname{num.val}, C_1.type); \}
                                                                           Btype=real \{a\} Ctype=real \{a\} T type=int i
                                                                                                                                        id; \{a\}D
             C.width = num.val * C_1.width; 
                                                                                                                            width=4
                                                                             width=8
                                                                                                    width=8
                          offset = 12
                                                                     real {a}
                                                                                                             B type=int\{a\}C_{type}=int\{a\} \varepsilon
                           t = int
                           w=4
```

## 例:数组类型表达式"int[2][3]"的语法制导翻译

```
type=array(2, array(3, int))
\bigcirc P \rightarrow \{ offset = 0 \} D
                                                                                              width=24
2D \rightarrow T \text{ id}; \{ enter( id.lexeme, T.type, offset ); \}
         offset = offset + T.width; D
                                                                                                     C type=array(2, array(3,int)
                                                                               R type=int\{a\}
\mathfrak{D} \rightarrow \varepsilon
                                                                                                         width=24
                                                                                  width=4
(4)T \rightarrow B \quad \{ t = B.type; w = B.width; \}
          C \in T.type = C.type; T.width = C.width; 
                                                                                             [ num ] C^{type=array}(3,int)
                                                                                                                                              {a}
                                                                            int \{a\}
\textcircled{5}T \rightarrow \uparrow T_1 \{ T.type = pointer(T_1.type); T.width = 4; \}
                                                                                                               width=12
\textcircled{6}B → int { B.type = int; B.width = 4; }
7B \rightarrow \text{real} \{ B.type = real; B.width = 8; \}
                                                                                                                 Ctype=int
                                                                                                     num ]
                                                                                                                                      {a}
\otimes C \rightarrow \varepsilon \ \{ C.type=t; C.width=w; \}
                                                                                                                    width=4
\mathfrak{D}C \to [\operatorname{num}]C_1 \{ C.type = \operatorname{array}(\operatorname{num.val}, C_1.type) \}
                                                                           t = int
            C.width = num.val * C_1.width; 
                                                                                                                       {a}
                                                                           w=4
```

对于嵌套建构,可能出现相同的名字,进入嵌套结构后就push一个新的符号表

