

Capítol 4 Anàlisi Semàntica

- 4.1 Tenim un llenguatge amb funcions, procediments, variables i constants. A les funcions i procediments els hi podem passar paràmetres per valor. Els tipus de dades possibles són: int, float, char, apuntador (es possible apuntador a funció i procediment), struct, union, array. Expressar l'estructura de dades utilitzada per representar el significat dels símbols del llenguatge.
- 4.2 Expressar el control semàntic de les regles BNF <factor> <terme>, <expressio_simple>, <expressio>.
- 4.3 Expressar l'anàlisi semàntica d'un RECORD de PASCAL sense CASE.
- 4.4 Expressar l'anàlisi semàntica de la llista de paràmetres d'un procedure de PASCAL a la seva declaració i crida. Es permet passar paràmetres per referència i valor.
- 4.5 Expressar l'anàlisi semàntica d'expressions d'accés a arrays, structs, unions i apuntadors de C per realitzar assignacions. Considereu els conceptes de l-value (valor amb adreça) i r-value (valor sense adreça) de C.
- 4.6 ¿Quan s'han de crear i eliminar els àmbits en les següents construccions?
¿Quins símbols conté cada àmbit?

Problema 4.1(cont.)

```
struct entr_TS
{ enum { simb,separador,tipus } classe;
  union
  { struct
    { char nom[32];
      enum { var,proc,func,const,param } classe;
      union { struct { struct entr_TS*tipus; } VAR;
              struct { struct entr_TS*tipus;
                        struct entr_TS*seg;
                      } PARAM;
              struct { struct entr_TS * tipus;
                        union { int I_valor;
                                float F_valor;
                                char C_valor;
                              } valor;
                        } CONST;
              struct { struct entr_TS* tipus_ret;
                        int num_param;
                        struct entr_TS* param;
                      } FUNCIO;
              struct { int num_param;
                        struct entr_TS* param;
                      } PROC;
            } s;
  } SIMB;
```

Problema 4.1(cont.)

struct

```
{ enum {integer, real, character, camp, apuntador, array, struct-union-record  
      } classe;
```

```
union
```

```
{ struct { struct entr_TS * tipus;  
          } APUNT;
```

```
  struct{ struct entr_TS* tipus;  
          int min,max;  
          } ARRAY;
```

```
  struct{ char nom[32];  
          struct entr_TS *camp;  
          } STRUCT-UNION-RECORD;
```

```
  struct{ char nom[32];  
          struct entr_TS *tip_camp;  
          struct entr_TS *seg_camp;  
          } CAMP;
```

```
  }t;
```

```
} TIPUS;
```

```
}c;
```

```
} SIMBOL;
```