

Declarations

Declaration of Variables

```
VariableElement: IdentList COLONSY Type SEMICOLONY  
                {vardecl($1, $3);} ;
```

```
Type           : SimpleType  
                | ArrayType  
                | RecordType ;
```

```
SimpleType     : IDENTSY  
                {$$ = typename(qualident($1));} ;
```

Identifier List

```
IdentList : IDENTSY  
          { $$ = identlist($1, NULL); }  
          | IDENTSY COMMASY IdentList  
          { $$ = identlist($1, $3); }  
          ;
```

Declared Constants

```
ConstElement : IDENTSY EQSY ConstExpression  
              SEMICOLONY  
              { consdecl($1, $3); }  
              ;
```

```
typedef struct cons_info CONS;
```

```
struct cons_info  
{ TYPE *cons_type;  
  int cons_value;  
}; /* cons_info */
```

Constant Expressions

```
ConstExpression : ConstExpression ORSY ConstExpression
                 {$$ = evalcons($1, OrOp, $3);}
                 | ConstExpression ANDSY ConstExpression
                 {$$ = evalcons($1, AndOp, $3);}
                 | NOTSY ConstExpression
                 {$$ = evalcons($2, NotOp, NULL);}
                 ...
                 | ConstExpression LESY ConstExpression
                 {$$ = evalcons($1, DivOp, $3);}
                 ...
                 | ConstExpression MODSY ConstExpression
                 {$$ = evalcons($1, ModOp, $3);}
                 | SUBSY ConstExpression %prec UNMINUSSY
                 {$$ = evalcons($2, MinOp, NULL); }
                 | LEFTPARENSY ConstExpression RIGHTPARENSY
                 {$$ = $2;}
                 | INTCONSTSY
                 {$$ = makeintcons($1);}
                 ...
                 | IDENTSY
                 {$$ = getcons(qualident($1));}
                 ;
```

CS 5300 - SJAllan

5

String Data Structure

```
typedef struct str_list STR;
```

```
struct str_list
{ char *str_ptr;
  STR *str_next;
}; /* str_list */
```

CS 5300 - SJAllan

6

Strings – Linked List

```
CONS *makestrcons (char *str)
{ CONS *cons_temp;
  STR *str_temp;
  cons_temp = (CONS *)malloc(sizeof(CONS));
  cons_temp->cons_type = str_type;
  cons_temp->cons_value = sdatasize;
  sdatasize += strlen(str) + 1;
  str_temp = (STR *)malloc(sizeof(STR));
  if (str_head == NULL)
    str_head = str_end = str_temp;
  else
    str_end->str_next = str_temp;
  str_end = str_temp;
  str_temp->str_ptr = str;
  return(cons_temp);
} /* makestrcons */
```

CS 5300 - SJAllan

7

TypeElement

```
TypeElement : IDENTSY EQSY Type SEMICOLONS  
             {typeddecl($1, $3);} ;
```

CS 5300 - SJAllan

8

SubrangeType

```
SubrangeType : LEFTBRACESY ConstExpression  
              COLONSY ConstExpression  
              RIGHTBRACESY  
              { $$ = subrange($2, $4); }  
              ;
```

RecordType

```
RecordType : RECORDSY FieldListSequence ENDSY  
            { $$ = recordtype($2); }  
            ;
```

FieldLists

```
FieldListSequence : FieldList
  { $$ = fieldlistsequence($1, NULL); }
| FieldListSequence SEMICOLONY FieldList
  { $$ = fieldlistsequence($1, $3); }
;

FieldList : IdentList COLONY Type
  { givetype($1, $3); $$ = $1; }
| /* empty */
  { $$ = NULL; }
;
```

ArrayType

```
ArrayType : ARRAYSY SubrangeType OFSY Type
  { $$ = arraytype($2, $4); }
;
```

ProcedureHeading

```
ProcedureHeading : PROCEDURESY ProcedureName  
                  FormalParameters  
                  {$$ = procedureheading($2, $3);}  
                  ;
```

```
ProcedureName    : IDENTSY  
                  {$$ = procedurename($1);}  
                  ;
```

FunctionHeading

```
FunctionHeading : FUNCTIONSY ProcedureName  
                FormalParameters COLONSY Type  
                {$$ = functionheading($2, $3, $5);}  
                ;
```

Formal Parameters

```
FormalParameters : LEFTPARENSEY OptFPSectionList RIGHTPARENSEY
                  { $$ = $2; }
                  ;
OptFPSectionList : FPSectionList
                  | /* empty */
                  { $$ = NULL; }
                  ;
FPSectionList    : FPSection
                  { $$ = $1; }
                  | FPSectionList SEMICOLONSEY FPSection
                  { $$ = fpsectionlist($1, $3); }
                  ;
FPSection        : VARSY IdentList COLONSEY SimpleType
                  { $$ = fpsection(RParameter, $2, $4); }
                  | IdentList COLONSEY SimpleType
                  { $$ = fpsection(VParameter, $1, $3); }
                  ;
```