PRÁCTICA 1: DISEÑO BNF

# <PROG>::=<DEC1><DEC2> | <DEC2>

# <DEC1>::=<INCS\_S><DEFS\_S><TIPOS\_S> |<DEFS\_S><TIPOS\_S> | <INCS\_S><TIPOS\_S> | <INCS\_S><DEFS\_S> | <INCS\_S> | <DEFS\_S> | <TIPOS\_S>

# <DEC2>::=<vars\_s><procS\_s><MAIN>| <VARS\_S><MAIN> | <PROCS\_S><MAIN> | <MAIN>

# <INCS\_S>::=<incs\_s><incs>|<INCS>

# <defs\_s>::=<DEFS\_S><DEFS>| <DEFS>

# <TIPOS\_S>::=<TIPOS\_S><TIPOS>|<TIPOS>

# <VARS\_S>::=<VARS\_S><VARS>|<VARS>

# <PROCS\_S>::=<PROCS\_S><PROC> | <PROC>

<INCS>::=#include<CABECERAS>

<CABECERAS>::= cab1 | cab2

<DEFS>::=#define alfnum

<TIPOS>::=typedef<TIPO>alf;

<TIPO>::=int | float | char | set | alf

<VARS>::=tipo <VARIABLES> ;

<VARIABLES>::=<VARIABLES>,<VARIABLES\_S>| <VARIABLES\_S>

<VARIABLES\_S>::=<VARIABLE> | <VARIAB LE> = <EXPRESION>

<VARIABLE>::= alf | alf [ num ]

<PROC>::=voidalf ( <PARAMS> ) <CUERPO> | voidalf () <CUERPO>

<PARAMS>::=<PARAMS>, <TIPO>alf | <TIPO>alf

<SETENCIA>::=<SWITCH> | <IF> | <WHILE> | <IN> | <OUT> | <LLAMADA\_PROC> | <LLAMADA\_CONJUNTO>; | <EXPRESION>;

<SWITCH>::= switch ( alf ) { <CASOS\_S><CASO\_DEFECTO> } | switch ( alf ) { <CASO\_DEFECTO> } | switch ( alf ) { <CASOS\_S> } | switch ( alf ) { }

<CASOS\_S>::=<CASOS\_S> case <LITERAL> :<SENTENCIAS>break; | case <LITERAL> :<SENTENCIAS>break; | <CASOS\_S> case <LITERAL> : break; | case <LITERAL> : break;

<LITERAL>::=num|conscar

<SENTENCIAS>::=<SENTENCIAS><SENTENCIA> |<SENTENCIA>

<CASO\_DEFECTO>::= default: <SENTENCIAS> | default:

<BLOQUE>::= { <SENTENCIAS> } | <SENTENCIA> | { }

<IF>::=if ( <EXPRESION> )<BLOQUE>else<BLOQUE> | if ( <EXPRESION> ) <BLOQUE>

<EXPRESION>::=<EXPRESION>+<EXPRESION> |<EXPRESION>-<EXPRESION> |<EXPRESION>\*<EXPRESION> | <EXPRESION>/<EXPRESION> | <EXPRESION>&&<EXPRESION> |<EXPRESION> || <EXPRESION> |<EXPRESION>>=<EXPRESION> |

<EXPRESION><= <EXPRESION> | <EXPRESION>><EXPRESION> | <EXPRESION><<EXPRESION> | <EXPRESION>!=<EXPRESION> |<EXPRESION>==<EXPRESION> |<NUMERO> | ( <EXPRESION>) | !<EXPRESION> | <ALF> | <REAL> | <CONSCAR> | <CREA\_CONJUNTO> | <LLAMADA\_DIFERENCIA> | < LLAMADA\_UNION> | <LLAMADA\_COMPLEMENTARIO> |<LLAMADA\_INTERSECCION> | <LLAMADA\_SIVACIO> | <LLAMADA\_LENGTH> | <ALF> = <EXPRESION>

<WHILE>::= while ( <EXPRESION> )<BLOQUE> | while (<EXPRESION>);

<OUT>::=print (texto\_out,<IDS> );| print (texto\_out );

<IDS>::=<IDS>,alf | alf

<IN>::=scanf ( op\_inout ,alf );

<LLAMADA\_PROC>::=alf (<PARAMS\_LLAMADA> ); | alf ();

<PARAMS\_LLAMADA>::=<PARAMS\_LLAMADA>, <EXPRESION> | <EXPRESION>

<MAIN>::=voidmain ( )<CUERPO>

<CUERPO>::={<VARS\_S><SENTENCIAS>} | { < SENTENCIAS> } | {<VARS\_S>} | { }

<LLAMADA\_CONJUNTO>::=<INSERTA\_CONJUNTO> | <SACA\_CONJUNTO> | <DESTRUYE\_CONJUNTO>

<CREA\_CONJUNTO>::=create ( )

<INSERTA\_CONJUNTO>::=insert (<EXPRESION> , alf)

<SACA\_CONJUNTO>::=extract (<EXPRESION> , alf)

<DESTRUYE\_CONJUNTO>::=delete ( alf )

<LLAMADA\_DIFERENCIA>::=minus( <EXPRESION> , <EXPRESION> )

<LLAMADA\_UNION>::=union( <EXPRESION> , <EXPRESION> )

<LLAMADA\_INTERSECCION>::=intersection ( <EXPRESION> , <EXPRESION> )

<LLAMADA\_COMPLEMENTARIO>::=neg ( <EXPRESION> )

<LLAMADA\_SIVACIO>::=empty( alf)

<LLAMADA\_LENGTH>::=length (alf )