Un subconjunto de la gramática de C++ en notación BNF

<unidad traducción> ::= λ | <declaraciones> <unidad traducción>

<declaraciones> ::= <especificador de tipo> <ident> <especificador declaración>

<especificador de tipo> ::= **void** | **char** | **int** | **float**

<especificador declaración> ::= <definición de función> | <declaración de variable>

<definición de función> ::= **(** <lista decl parámetros>**)** <proposición compuesta>

<lista decl parámetros> ::= λ | <declaración parámetros> <resto lista decl par>

<resto lista decl par> ::= λ | **,** <declaración parámetros> <resto lista decl par>

<declaración parámetros> ::= <especificador de tipo> <oprefopcional> <ident> <arregloopcional>

<oprefopcional> ::= λ | **&**

<arregloopcional> ::= λ | **[ ]**

<declaración de variable> ::= <declarador init> <lista decl init> **;**

<lista decl init> ::= λ | **,** <resto lista decl init>

<resto lista decl init> ::= <ident> <declarador init> <lista decl init>

<declarador init> ::= λ | **=** <opción de constante>

<opción de constante> ::= <constante> | **[** <límite opcional > **]** <lista opcional>

<límite opcional> ::= λ | <constante>

<lista opcional> ::= λ | **= {** <lista inicializadores> **}**

<lista inicializadores> ::= <constante> <resto lista inic>

<resto lista inic> ::= λ | **,** <constante> <resto lista inic>

<proposición compuesta> ::= **{** <lista declaración> <lista proposición> **}**

<lista declaración> ::= λ | <declaración> <lista declaración>

<declaración> ::= <especificador de tipo> <resto lista decl init> **;**

<lista proposición> ::= λ | <proposición> <lista proposición>

<proposición> ::= <proposición selección> | <proposición iteración> |
<proposición expresión> | <proposición compuesta> |
<proposición ent/sal> | <proposición retorno>

<proposición selección> ::= **if (** <expresión> **)** <proposición> <else opcional>

<else opcional> ::= λ | **else** <proposición>

<proposición iteración> ::= **while (** <expresión> **)** <proposición>

<proposición retorno> ::= **return** <expresión> **;**

<proposición expresión> ::= **;** | <expresión> **;**

<proposición ent/sal> ::= **cin >>** <variable> <resto prop in> **;** |
cout << <expresión> <resto prop out> **;**

<resto prop in> ::= λ | **>>** <variable> <resto prop in>

<resto prop out> ::= λ | **<<** <expresión> <resto prop out>

<expresión> ::= <expresión simple> <resto expresión>

<resto expresión> ::= λ | <op relacional> <expresión simple> <resto expresión>
| **=** <expresión simple> <resto expresión>

<op relacional> ::= **=** | **!=** | **==** | **<** | **<=** | **>=** | **>**

<expresión simple> ::= <operador opcional> <término> <resto expresión simple>

<operador opcional> ::= λ | **+** | **-**

<resto expresión simple> ::= λ | <operador> <término> <resto expresión simple>

<operador> ::= **+** | **-** | **||**

<término> ::= <factor> <resto término>

<resto término> ::= λ | <opermul> <factor> <resto término>

<opermul> ::= ***** | **/** | **&&**

<factor> ::= <variable> |
<constante> |
! <expresión> |
(<expresión> **)** |
<invocación función> |
<cte-string>

<variable> ::= <ident> <resto opcional>

<resto opcional> ::= λ | **[** <expresión> **]**

<invocación función> ::= <ident> **(** <lista expresiones> **)**

<lista expresiones> ::= λ | <expresión> <resto lista expr>

<resto lista expr> ::= λ | **,** <expresión> <resto lista expr>

<constante> ::= <cte-num> | <cte-char>

<cte-num> ::= <número> <resto cte opcional>

<resto cte opcional> ::= λ | **.** <número>

<número> ::= <dígito> <resto número>

<resto número> ::= λ | <dígito> <resto número>

<cte-char> ::= **‘** <ascii> **’**

<cte-string> ::= **“** <cadena> **”**

<cadena> ::= λ | <ascii> <cadena>

<dígito> ::= **0** | **1** | … | **9**

<ascii> ::= <letra> | <dígito> | ***** | **¿** | **^** | ….

<ident> ::= <letra> <resto ident>

<resto ident> ::= λ | <letra> <resto ident> | <dígito> <resto ident>

<letra> ::= **a** | … | **z** | **A** | …. | **Z**