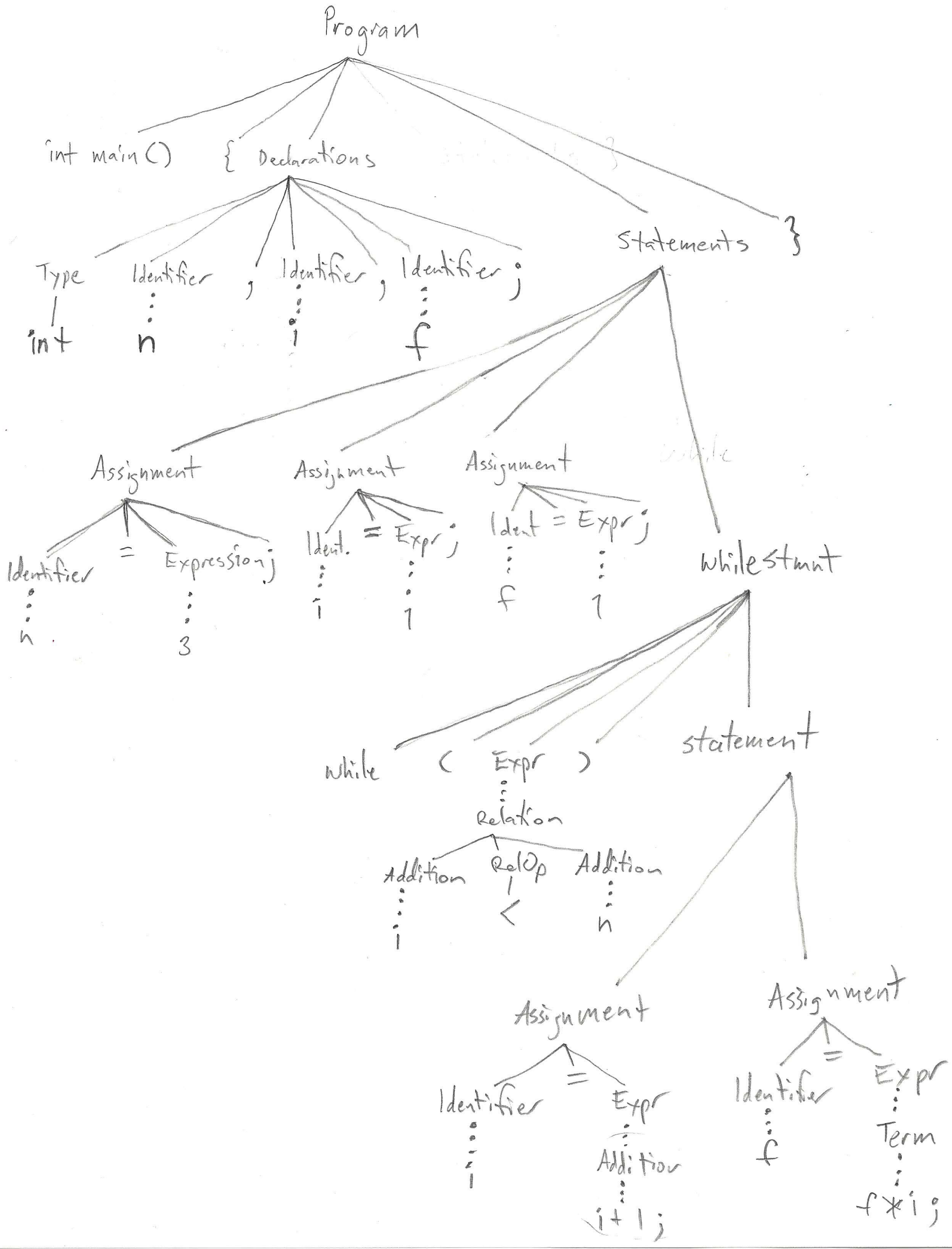
```
Clite-student
$ java Lexer factorial.cpp
Tokens:
int
main
(
)
{
int
Identifier     n
,
Identifier     i
,
Identifier     f
;
Identifier     n
=
IntLiteral     3
;
Identifier     i
=
IntLiteral     1
;
Identifier     f
=
IntLiteral     1
;
while
(
Identifier     i
<
Identifier     n
)
{
Identifier     i
=
Identifier     i
+
IntLiteral     1
;
Identifier     f
=
Identifier     f
*
Identifier     i
;
}
}
```

Clite-student

```
$ java Parser factorial.cpp
Output:
Program (abstract syntax):

  Declarations:  {<int: n> , <int: i> , <int: f>}

  Statements:
    Assignment:
      Target:
        Variable: n
      Source:
        IntVal: 3
    Assignment:
      Target:
        Variable: i
      Source:
        IntVal: 1
    Assignment:
      Target:
        Variable: f
      Source:
        IntVal: 1
    Loop:
      Binary Op:  <
        Variable: i
        Variable: n
        Assignment:
          Target:
            Variable: i
          Source:
            Binary Op:  +
              Variable: i
              IntVal: 1
        Assignment:
          Target:
            Variable: f
          Source:
            Binary Op:  *
              Variable: f
              Variable: i
```


Program
int main()
{
Declarations
Statements
}
Type
Identifier
,
Identifier
,
Identifier
;
int
n
i
f
Assignment
Identifier
=
Expression
;
n
3
Assignment
Ident.
=
Expr
;
i
1
Assignment
Ident
=
Expr
;
f
1
whilestmnt
while
(
Expr
)
Relation
Addition
RelOp
Addition
i
<
n
statement
Assignment
Identifier
=
Expr
Addition
i
i+1;
Assignment
Identifier
=
Expr
f
Term
f*i;