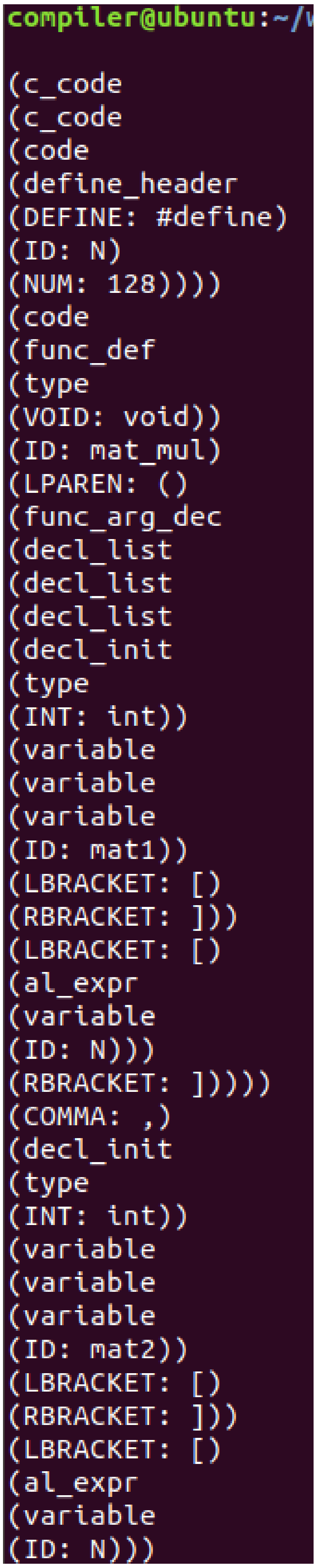
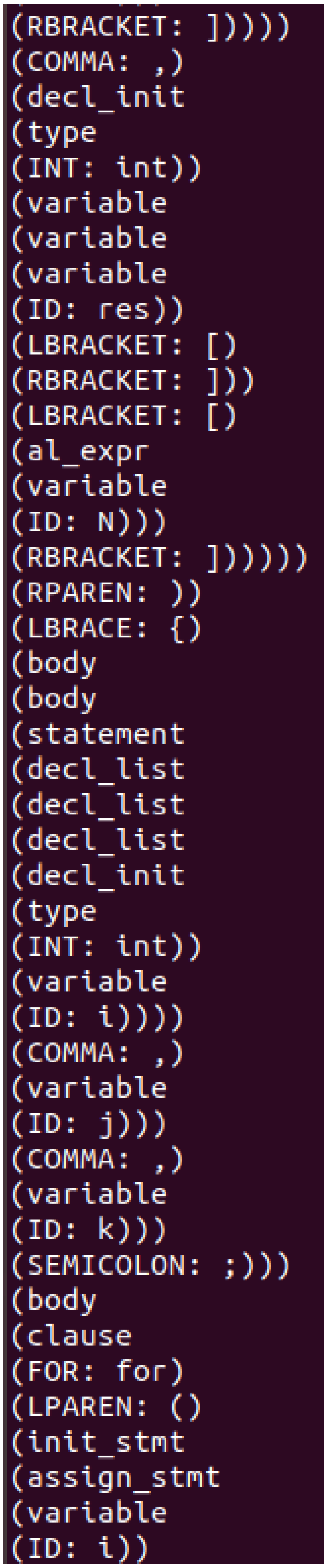
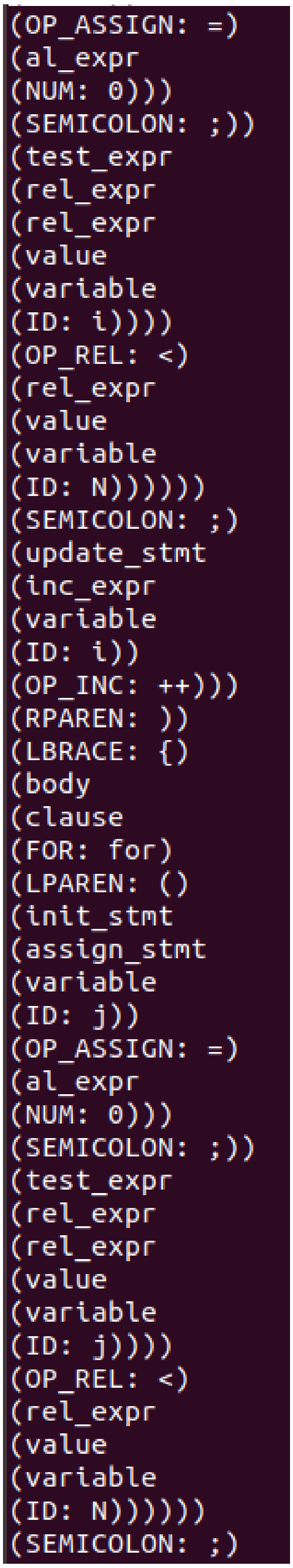


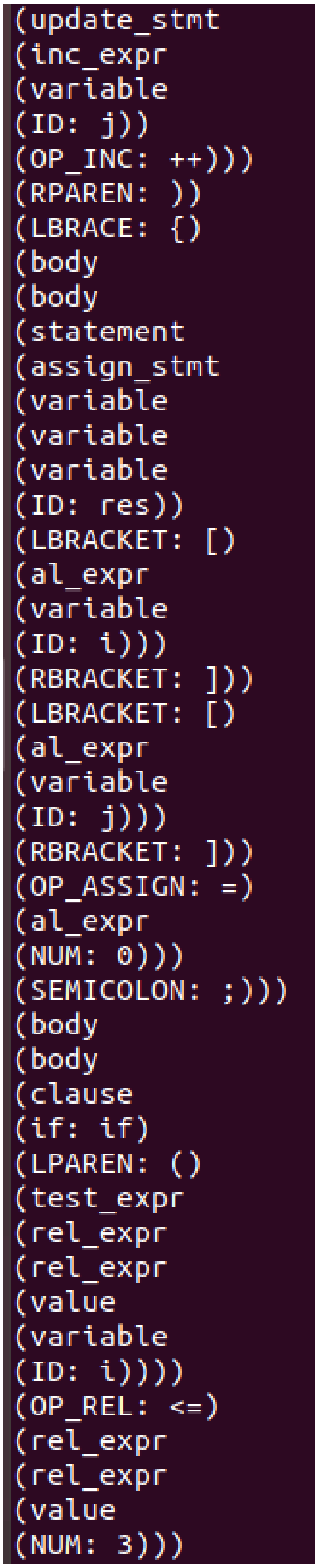
```
compiler@ubuntu:~/

(c_code
(c_code
(code
(define_header
(DEFINE: #define)
(ID: N)
(NUM: 128))))
(code
(func_def
(type
(VOID: void))
(ID: mat_mul)
(LPAREN: ()
(func_arg_dec
(decl_list
(decl_list
(decl_list
(decl_init
(type
(INT: int))
(variable
(variable
(variable
(ID: mat1))
(LBRACKET: [)
(RBRACKET: ]))
(LBRACKET: [)
(al_expr
(variable
(ID: N)))
(RBRACKET: ]))))
(COMMA: ,)
(decl_init
(type
(INT: int))
(variable
(variable
(variable
(ID: mat2))
(LBRACKET: [)
(RBRACKET: ]))
(LBRACKET: [)
(al_expr
(variable
(ID: N)))
```

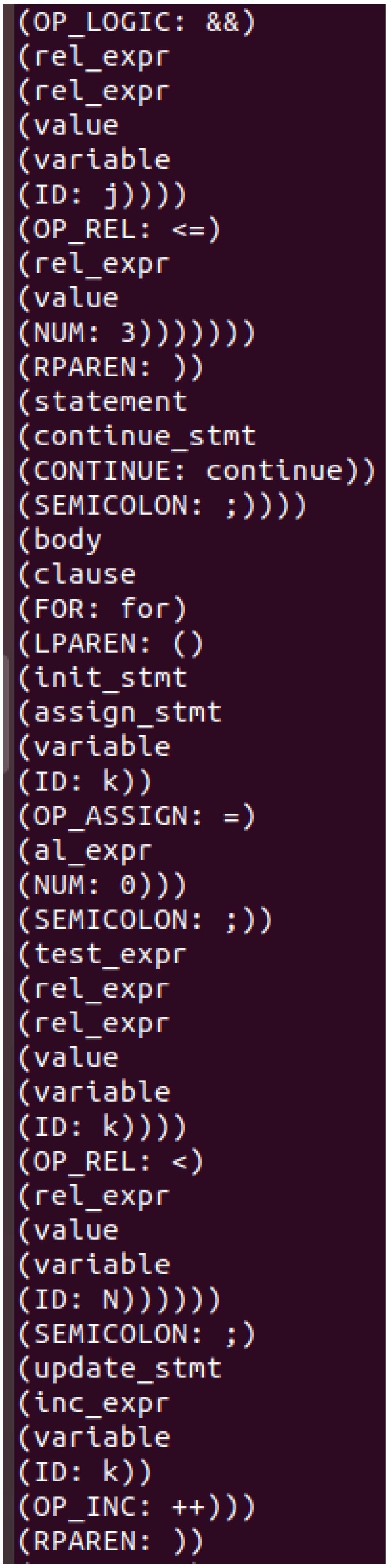

```
(RBRACKET: ]))))
(COMMA: ,)
(decl_init
(type
(INT: int))
(variable
(variable
(variable
(ID: res))
(LBRACKET: [)
(RBRACKET: ]))
(LBRACKET: [)
(al_expr
(variable
(ID: N)))
(RBRACKET: ])))))
(RPAREN: ))
(LBRACE: {)
(body
(body
(statement
(decl_list
(decl_list
(decl_list
(decl_init
(type
(INT: int))
(variable
(ID: i))))
(COMMA: ,)
(variable
(ID: j)))
(COMMA: ,)
(variable
(ID: k)))
(SEMICOLON: ;)))
(body
(clause
(FOR: for)
(LPAREN: ()
(init_stmt
(assign_stmt
(variable
(ID: i))
```

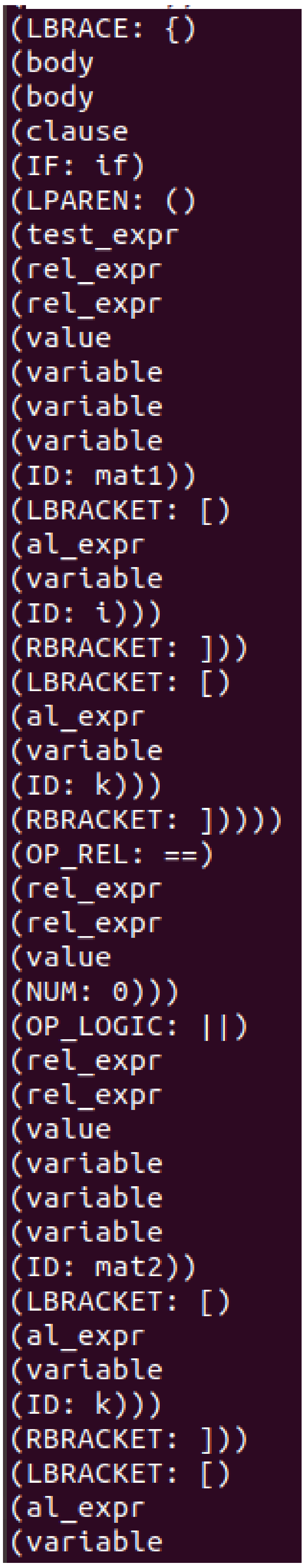


```
(OP_ASSIGN: =)
(al_expr
(NUM: 0)))
(SEMICOLON: ;))
(test_expr
(rel_expr
(rel_expr
(value
(variable
(ID: i))))
(OP_REL: <)
(rel_expr
(value
(variable
(ID: N))))))
(SEMICOLON: ;)
(update_stmt
(inc_expr
(variable
(ID: i))
(OP_INC: ++)))
(RPAREN: ))
(LBRACE: {)
(body
(clause
(FOR: for)
(LPAREN: ()
(init_stmt
(assign_stmt
(variable
(ID: j))
(OP_ASSIGN: =)
(al_expr
(NUM: 0)))
(SEMICOLON: ;))
(test_expr
(rel_expr
(rel_expr
(value
(variable
(ID: j))))
(OP_REL: <)
(rel_expr
(value
(variable
(ID: N))))))
(SEMICOLON: ;)
```

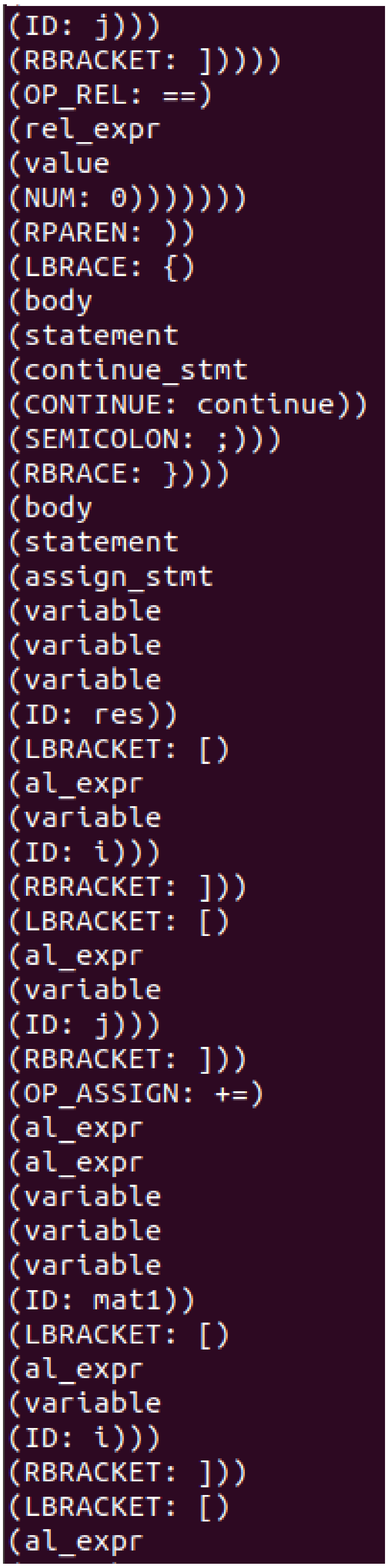


```
(update_stmt
(inc_expr
(variable
(ID: j))
(OP_INC: ++)))
(RPAREN: ))
(LBRACE: {)
(body
(body
(statement
(assign_stmt
(variable
(variable
(variable
(ID: res))
(LBRACKET: [)
(al_expr
(variable
(ID: i)))
(RBRACKET: ]))
(LBRACKET: [)
(al_expr
(variable
(ID: j)))
(RBRACKET: ]))
(OP_ASSIGN: =)
(al_expr
(NUM: 0)))
(SEMICOLON: ;)))
(body
(body
(clause
(if: if)
(LPAREN: ()
(test_expr
(rel_expr
(rel_expr
(value
(variable
(ID: i))))
(OP_REL: <=)
(rel_expr
(rel_expr
(value
(NUM: 3)))
```

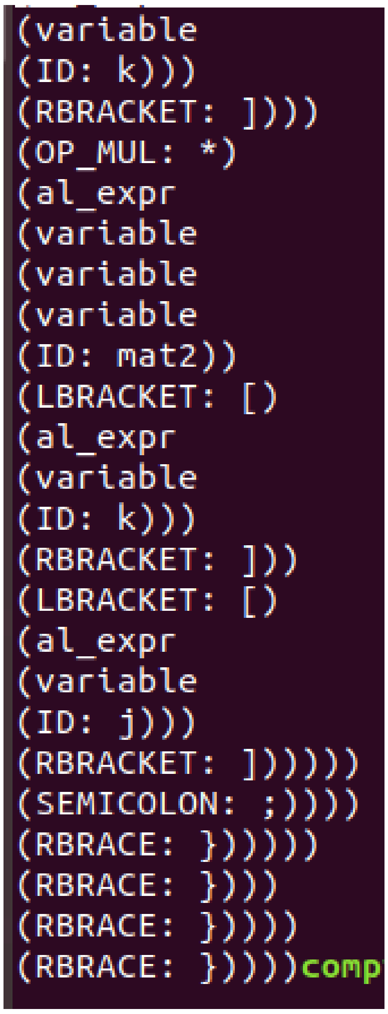

```
(OP_LOGIC: &&)
(rel_expr
(rel_expr
(value
(variable
(ID: j))))
(OP_REL: <=)
(rel_expr
(value
(NUM: 3)))))))
(RPAREN: ))
(statement
(continue_stmt
(CONTINUE: continue))
(SEMICOLON: ;))))
(body
(clause
(FOR: for)
(LPAREN: ()
(init_stmt
(assign_stmt
(variable
(ID: k))
(OP_ASSIGN: =)
(al_expr
(NUM: 0)))
(SEMICOLON: ;))
(test_expr
(rel_expr
(rel_expr
(value
(variable
(ID: k))))
(OP_REL: <)
(rel_expr
(value
(variable
(ID: N))))))
(SEMICOLON: ;)
(update_stmt
(inc_expr
(variable
(ID: k))
(OP_INC: ++)))
(RPAREN: ))
```



```
(LBRACE: {)
(body
(body
(clause
(IF: if)
(LPAREN: ()
(test_expr
(rel_expr
(rel_expr
(value
(variable
(variable
(variable
(ID: mat1))
(LBRACKET: [)
(al_expr
(variable
(ID: i)))
(RBRACKET: ]))
(LBRACKET: [)
(al_expr
(variable
(ID: k)))
(RBRACKET: ]))))
(OP_REL: ==)
(rel_expr
(rel_expr
(value
(NUM: 0)))
(OP_LOGIC: ||)
(rel_expr
(rel_expr
(value
(variable
(variable
(variable
(ID: mat2))
(LBRACKET: [)
(al_expr
(variable
(ID: k)))
(RBRACKET: ]))
(LBRACKET: [)
(al_expr
(variable
```



```
(ID: j)))
(RBRACKET: ]))))
(OP_REL: ==)
(rel_expr
(value
(NUM: 0)))))))
(RPAREN: ))
(LBRACE: {)
(body
(statement
(continue_stmt
(CONTINUE: continue))
(SEMICOLON: ;)))
(RBRACE: })))
(body
(statement
(assign_stmt
(variable
(variable
(variable
(ID: res))
(LBRACKET: [)
(al_expr
(variable
(ID: i)))
(RBRACKET: ]))
(LBRACKET: [)
(al_expr
(variable
(ID: j)))
(RBRACKET: ]))
(OP_ASSIGN: +=)
(al_expr
(al_expr
(variable
(variable
(variable
(ID: mat1))
(LBRACKET: [)
(al_expr
(variable
(ID: i)))
(RBRACKET: ]))
(LBRACKET: [)
(al_expr
```



```
(variable
(ID: k)))
(RBRACKET: ])))
(OP_MUL: *)
(al_expr
(variable
(variable
(variable
(ID: mat2))
(LBRACKET: [)
(al_expr
(variable
(ID: k)))
(RBRACKET: ]))
(LBRACKET: [)
(al_expr
(variable
(ID: j)))
(RBRACKET: ])))))
(SEMICOLON: ;))))
(RBRACE: }))))))
(RBRACE: })))
(RBRACE: }))))
(RBRACE: }))))comp
```