Cvv Compiler

Written on C++17.
Based on C.
32-bit architecture. Intel ASM.

Original series: https://norasandler.com/

Tokens:

Name	Value	Index
O_PRN	(	0
C_PRN	)	1
O_BRACE	{	2
C_BRACE	}	3
SEMI	;	4
KEYWORD	int	5
RETURN	return	6
IDENTIFIER	Names of variables, functions, etc.	7
I_NUM	Integer numbers	8
ENDOFFILE	EOF	9
ANEG	-	10
COMPLEMENT	~	11
LNEG	!	12
ADD	+	13
MUL	*	14
DIV	/	15
AND	&&	16
OR	||	17
EQU	==	18
NEQU	!=	19
LESS	<	20
LESSEQU	<=	21
GREAT	>	22
GREATEQU	>=	23
ASSIGN	=	24
QUESTION	?	25
COLON	:	26
IF	if	27
ELSE	else	28
FOR	for	29
DO	do	30
WHILE	while	31
BREAK	break	32
COMA	,	33

Done:

- No Cyrillic
- Identifiers may contain only ASCII symbols, underscore and numbers. Can start with number
- Only /*...*/ comments
- Return int
- Unary operations

Negation	-
Bitwise complement	~
Logical negation	!

- Binary operations:

Addition	+
Subtraction	-
Multiplication	*
Division	/
Logical AND	&&
Logical OR	||
Equal to	==
Not equal to	!=
Less than	<
Greater than	>
Less than or equal to	<=
Greater than or equal to	>=

- Local variables
- Conditionals:

If-else
Ternary operator

- Compound statements. Examples:

```
// here is the outer scope
{
    // here is the inner scope
    int foo = 2;
}
// now we're back in the outer scope
foo = 3; // ERROR - foo isn't defined in this scope!
```

```
int a = 2;
{
    a = 4; // this is okay
}
return a; // returns 4 - changes made inside the inner scope are reflected here
```

```
int foo = 0;
{
    int foo; // this is a TOTALLY DIFFERENT foo, unrelated to foo from earlier
    foo = 2; // this refers to the inner foo; outer foo is inaccessible
}
return foo; //this will return 0 - it refers to the original foo, which is
unchanged
```

```
int foo = 0;
{
    foo = 3; //changes outer foo
    int foo = 4; //defines inner foo, shadowing outer foo
}
return foo; //returns 3
```

- Loops:

For
While
Do-while
Break

To do:

- Return other types
- Other binary operations:

Modulo	%
Bitwise AND	&
Bitwise OR	|
Bitwise XOR	^
Bitwise shift left	<<
Bitwise shift right	>>

- Modifiers like short, long or unsigned.
- Storage-class specifiers like static
- Type qualifiers like const
- Statements like int a, b;
- Make main return 0 even if the return statement is missing
- Compound assignment operators:

+=
-=
/=
*=
%=
<<=
>>=
&=
|=
^=
++
--

- Continue keyword
- Functions
- Global variables

Operator precedence (high - low):

*, /
+, -
<, >, <=, >=
==, !=
&&
||

Grammar:

```
S             -> F
F             -> int ident () { {{BLOCK_ITEM}} }
BLOCK_ITEM    -> STAT | VARDECL
STAT          -> RET | VARDECL | VARASSIGN | IF_ELSE | {{ {BLOCK_ITEM} }} | FOROPT | FORDECL | WHILE | DO | BREAK
FOROPT        -> for ( EXPROPT; EXPROPT; EXPROPT ) STAT
FORDECL       -> for ( VARDECL  EXPROPT; EXPROPT ) STAT
WHILE         -> while ( EXPR ) STAT
DO            -> do STAT while EXPR;
BREAK         -> break;
IF_ELSE       -> if ( EXPR ) STAT [[else STAT]]
RET           -> return EXPR;
VARDECL       -> int ident [[ = EXPR ]] ;
VARASSIGN     -> EXPROPT ;
EXPROPT       -> EXPR | " "
EXPR          -> ident = EXPR | CONDITIONAL
CONDITIONAL   -> LOGOR [[? EXPR : CONDITIONAL]]
LOGOR         -> LOGAND {{ || LOGAND }}
LOGAND        -> EQU {{ && EQU }}
EQU           -> RELATIONAL {{ = | ==          RELATIONAL}}
RELATIONAL    -> ADDITIVE {{ < | > | <= | >=   ADDITIVE }}
ADDITIVE      -> TERM {{ + | - TERM }}
TERM          -> FACTOR {{ *|/  FACTOR }}
FACTOR              -> (EXPR) | UNOP FACTOR | iconst | ident
UNOP          -> ! | ~ | -
```

{{ ... }}	multiple repetition or absence
[[...]]	optional