Weak language draft

epoll-reactor

December 2021

Contents

1 Scope

This document describes requirements for implementation of weak programming language.

2 Lexical elements

2.1 Keywords

boolean	**break**	**char**
continue	**do**	**false**
for	**if**	**int**
return	**string**	**true**
void	**while**	

2.2 Operators and punctuators

`=`	`*=`	`/=`	`%=`	`+=`	`-=`
`<<=`	`>>=`	`&=`	`|=`	`^=`	`&&`
`|`	`^`	`&`	`==`	`!=`	`>`
`<`	`>=`	`<=`	`<<`	`>>`	`+`
`-`	`*`	`/`	`%`	`++`	`--`
`[`	`]`	`&`	`==`	`!=`	`>`
`(`	`)`	`{`	`}`		

3 Grammar summary

⟨*program*⟩ ::= ⟨*function-declaration*⟩ *

⟨*function-declaration*⟩ ::= ⟨*ret-type*⟩ ⟨*id*⟩ ***(*** ⟨*parameter-list-opt*⟩ ***)*** **{** ⟨*stmt*⟩ * **}**

⟨*ret-type*⟩ ::= ⟨*type*⟩
| ⟨*void-type*⟩

⟨*type*⟩ ::= ***int***
| ***char***
| ***string***
| ***boolean***

⟨*void-type*⟩ ::= ***void***

⟨*constant*⟩ ::= ⟨*integral-literal*⟩
| ⟨*floating-literal*⟩
| ⟨*string-literal*⟩
| ⟨*boolean-literal*⟩

```
⟨integral-literal⟩      ::= ⟨digit⟩ *

⟨floating-literal⟩      ::= ⟨digit⟩ * . ⟨digit⟩ *

⟨string-literal⟩        ::= " ( \x00000000-\x0010FFFF )* "

⟨boolean-literal⟩       ::= true
                          | false

⟨alpha⟩                 ::= a | b | ... | z | _

⟨digit⟩                 ::= 0 | 1 | ... | 9

⟨id⟩                    ::= ⟨alpha⟩ ( ⟨alpha⟩ | ⟨digit⟩ )*

⟨parameter⟩             ::= ⟨type⟩ ⟨id⟩

⟨parameter-list⟩        ::= ⟨parameter⟩ , ⟨parameter-list⟩
                          | ⟨parameter⟩

⟨parameter-list-opt⟩    ::= ⟨parameter-list⟩ | ϵ

⟨stmt⟩                  ::= ⟨selection-stmt⟩
                          | ⟨iteration-stmt⟩
                          | ⟨jump-stmt⟩
                          | ⟨expr⟩

⟨iteration-stmt⟩        ::= ⟨stmt⟩
                          | break;
                          | continue;

⟨selection-stmt⟩        ::= if ( ⟨expr⟩ ) { ⟨stmt⟩ * }
                          | if ( ⟨expr⟩ ) { ⟨stmt⟩ * } else { ⟨stmt⟩ * }

⟨iteration-stmt⟩        ::= for ( ⟨expr-opt⟩ ; ⟨expr-opt⟩ ; ⟨expr-opt⟩ ) { ⟨iteration-stmt⟩ *
                            }
                          | while ( ⟨expr⟩ ) { ⟨iteration-stmt⟩ * }
                          | do { ⟨iteration-stmt⟩ * } while ( ⟨expr⟩ )

⟨jump-stmt⟩             ::= return ⟨expr⟩? ;

⟨assignment-op⟩         ::= =
                          | *=
                          | /=
                          | %=
                          | +=
                          | -=
                      |   <<=
                      |   >>=
                      |   &=
                      |   |=
                      |   ^=

⟨expr⟩               ::= ⟨assignment-expr⟩

⟨expr-opt⟩           ::= ⟨expr⟩ | ϵ

⟨assignment-expr⟩    ::= ⟨logical-or-expr⟩
                      |   ⟨unary-expr⟩ ⟨assignment-op⟩ ⟨assignment-expr⟩

⟨logical-or-expr⟩    ::= ⟨logical-and-expr⟩
                      |   ⟨logical-or-expr⟩ || ⟨logical-and-expr⟩

⟨logical-and-expr⟩   ::= ⟨inclusive-or-expr⟩
                      |   ⟨logical-and-expr⟩ && ⟨inclusive-or-expr⟩

⟨inclusive-or-expr⟩  ::= ⟨exclusive-or-expr⟩
                      |   ⟨inclusive-or-expr⟩ | ⟨exclusive-or-expr⟩

⟨exclusive-or-expr⟩  ::= ⟨and-expr⟩
                      |   ⟨exclusive-or-expr⟩ ^ ⟨and-expr⟩

⟨and-expr⟩           ::= ⟨equality-expr⟩
                      |   ⟨and-expr⟩ & ⟨equality-expr⟩

⟨equality-expr⟩      ::= ⟨relational-expr⟩
                      |   ⟨equality-expr⟩ == ⟨relational-expr⟩
                      |   ⟨equality-expr⟩ != ⟨relational-expr⟩

⟨relational-expr⟩    ::= ⟨shift-expr⟩
                      |   ⟨relational-expr⟩ > ⟨shift-expr⟩
                      |   ⟨relational-expr⟩ < ⟨shift-expr⟩
                      |   ⟨relational-expr⟩ >= ⟨shift-expr⟩
                      |   ⟨relational-expr⟩ <= ⟨shift-expr⟩

⟨shift-expr⟩         ::= ⟨additive-expr⟩
                      |   ⟨shift-expr⟩ << ⟨additive-expr⟩
                      |   ⟨shift-expr⟩ >> ⟨additive-expr⟩

⟨additive-expr⟩      ::= ⟨multiplicative-expr⟩
                      |   ⟨additive-expr⟩ + ⟨multiplicative-expr⟩
                      |   ⟨additive-expr⟩ - ⟨multiplicative-expr⟩
```

```
⟨multiplicative-expr⟩   ::= ⟨unary-expr⟩
                          | ⟨multiplicative-expr⟩ * ⟨unary-expr⟩
                          | ⟨multiplicative-expr⟩ / ⟨unary-expr⟩
                          | ⟨multiplicative-expr⟩ % ⟨unary-expr⟩

⟨unary-expr⟩            ::= ⟨postfix-expr⟩
                          | ++ ⟨unary-expr⟩
                          | -- ⟨unary-expr⟩

⟨postfix-expr⟩          ::= ⟨primary-expr⟩
                          | ⟨postfix-expr⟩ [ ⟨expr⟩ ]
                          | ⟨postfix-expr⟩ ++
                          | ⟨postfix-expr⟩ --

⟨primary-expr⟩          ::= ⟨constant⟩
                          | ⟨id⟩
                          | ( ⟨expr⟩ )
```

4 Environment

4.1 Translation environment

The whole program must be placed in one file to simplify translation and linking (lack of it as such).