IFJ2020 Grammar rules – *by Poli && Coacher*

1. Terminals

ID, K_WORD, PLUS, MINUS, TIMES, DIV, EQ, L_EQ, G_EQ, LESS, GREAT, N_EQ, L_VINC(, R_VINCL), LEFT_BRAC{, RIGHT_BRAC}, IF, ELSE, FOR, IS, ASSIGN, STRING_LITERAL, INT_LITERAL, COMMA, DOUBLE_LITERAL, RETURN, PACKAGE_MAIN, Q_MARK (uvozovky), F_MAIN (povinná funkce main), TRUE, FALSE, EOF, EOL, FUNC(jen speciální K word..), ASSIGN, IF, ELSE, FOR, SEMICOLON

2. Non-Terminals

<prog>, <func>, <param>, <param_n>, <retval>, <retvals>, <body>, <f_retval>, <f_retvals>, <expr>, <func_call>, <call-params>, <if>, <for>, <bool_expr>, <for_init>

3. Rules

----------Funkce, tělo programu, definice funkcí a návratových paramatrů----------

a. <prog> *// jelikož v IFJ20 nejsou globalní promenne tak povolujeme na zacatku jen definice fci*
   i. <prog> -> PACKAGE_MAIN <func>
   ii. <prog> -> EOF
b. <func> *// definice funkce: func id (params) (retvals){ body } může následovat další funkce*
   i. <func>-> FUNC ID L_VINCL <param_n> R_VINCL <retvals> L_BRAC <body> <f_retvals> R_BRAC EOL<func>
   ii. <func> -> ε
c. <param_n> *// buď nejsou žádné parametry: () nebo (param)*
   i. <param_n> -> ε
   ii. <param_n> -> <param>
d. <param> *// buď jeden parametr i, nebo víc parametrů ii*
   i. <param> -> [INT, DOUBLE, STRING] ID
   ii. <param>-> [INT, DOUBLE, STRING] ID COMMA <param>
e. <retvals> *// návratové hodnoty nemusí být přítomny a nebo pokračuj na retval*
   i. <retvals> -> L_VINCL <retval> R_VINCL
   ii. <retvals> -> ε
f. <retval> *// buď jedna návratová hodnota nebo více*
   i. <retval> -> [INT, DOUBLE, STRING]
   ii. <retval>-> [INT, DOUBLE, STRING] COMMA <retval>
g. <f_retvals> *// návratové hodnoty uvnitř těla funkce, nemusí být nebo pokračuj na f_retval*
   i. <f_retval> -> ε
   ii. <f_retval> -> <f_retval>
h. <f_retval> *// návratové hodnoty mohou být jeden či více výrazů*
   i. <f_retval> -> <expr>
   ii. <f_retval> -> <expr> COMMA <f_retval>

----------Tělo funkce, cykly, aritemtické výrazy----------

i. <body> *// jsme v tělu funkce, buď přiřazujeme do proměnné výraz nebo volání funkce, také se může vyskytnout for, if nebo EOL*
   i. <body> -> ID ASSIGN <expr> EOL <body>

ii. <body> -> ε
iii. <body> -> ID ASSIGN <func-call> EOL <body>
iv. <body> -> <exp> EOL <body>
v. <body> -> <if> <body>
vi. <body> -> <for> <body>
vii. <body> ->EOL

j. <func-call> *// přiřazujeme z volání funkce*
   i. <func-call> -> ID L_VINCL <call_params> R_VINCL
k. <call-params> *// nemusí být žádné paramentry z volání funkce, jinak pokračuj na call-param*
   i. <call-params> -> ε
   ii. <call-params>-> <call-param>
l. <call-param>*// buď máme jeden parametr nebo více*
   i. <call-param> -> <expr>
   ii. <call-param> -> <expr> COMMA <call-param>
m. <if>*// if obsahuje bool podmínku zároky tělo, else další závorky a tělo*
   i. <if> -> IF <bool_expr> R_BRAC EOL <body> L_BRAC ELSE R_BRAC EOL <body> L_BRAC
n. <for> *// for (init; condition; after)*
   i. <for> -> for L_VINCL<for_init> SEMICOLON <bool_expr> SEMICOLON <expr> R_VINCL L_BRAC EOL <body> R_BRAC
o. <for_init> *// to co může být v první podmínce ve for*
   i. <for_init> -> [INT, DOUBLE, STRING] ID ASSIGN <expr>