nonterminal	first set	follow set	nullable	endable
INSTRUCTIONS	accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch (	; until accolade_fermante case default	no	yes
FINSTRUCTION	accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch (	; until accolade_fermante case default	yes	yes
AFFECTATION	id	;	no	no
Fid	deux_points egal left_arrow	;	no	no
FTERM	+ minus_sign	double_ampersand double_barre and orcomparator) virgule deux_points ;	yes	no
TERM	+ minus_sign id number boolean string call	double_ampersand double_barre and or + minus_sign comparator) virgule deux_points ;	no	no
FFACTEUR	* mult slash div pourcentage mod modulo increment decrement	double_ampersand double_barre and or + minus_sign comparator) virgule deux_points ;	yes	no
OPERATEURMULT	* mult slash div pourcentage mod modulo	id number boolean string call	no	no
OPERATEURSPECIAUX	increment decrement	double_ampersand double_barre and or + minus_sign comparator) virgule deux_points ;	no	no
APPEL_FONCTION	call	double_ampersand double_barre and or * mult slash div pourcentage mod modulo increment decrement + minus_sign comparator) virgule deux_points ;	no	no
APPEL_FONCTION_ARG	) id	double_ampersand double_barre and or * mult slash div pourcentage mod modulo increment decrement + minus_sign comparator) virgule deux_points ;	no	no

ARGUMENT$_1$	virgule	)	yes	no
RETURN	return	;	no	no
BOUCLE	for do repeat while	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	no	yes
FORLOOP_STATEMENT	for	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	no	yes
Ffor	(id	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	no	yes
F(	const let	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	no	yes
FVAR_DECLARATION	; deux_points	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	no	yes
F;	! not (+ minus_sign id number boolean string call	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	no	yes
FCONDITIONS$_1$	;	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	no	yes
F;$_1$	id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch (	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	no	yes
FINSTRUCTION$_1$	)	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	no	yes
F)$_3$	accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch (	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	no	yes
Fdeux_points$_1$	id	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	no	yes
Fid$_4$	)	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	no	yes
F)$_2$	accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch (	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	no	yes
Fid$_3$	in egal	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts	no	yes

		gets def function const let for do repeat while if switch ( ) deux_points		
WHILELOOP_STATEMENT	while	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	no	yes
Fwhile	(	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	no	yes
$F(_1$	! not (+ minus_sign id number boolean string call	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	no	yes
FCONDITIONS	)	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	no	yes
F)	accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch (	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	no	yes
DOWHILELOOP_STATEMENT	do repeat	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	no	yes
FCONDITION	double_ampersand double_barre and or	) ;	yes	no
CONDITIONS	! not (+ minus_sign id number boolean string call	) ;	no	no
INPUT_OUTPUT	print printf scanf input log fprintf fscanf fread fwrite write read puts gets	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	no	yes
ARGUMENT	id	)	no	no
FONCTION	def function	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	no	yes
FONCTION2	id	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	no	yes
PARAMETER	id	)	no	no
$PARAMETER_1$	virgule	)	yes	no
CONTROLE	if switch (	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	no	yes
IF	if	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	no	yes
Fif	(	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	no	yes
$F(_2$	(+ minus_sign id number boolean	; until accolade_fermante case default	no	yes

	string call	accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points		
$FCONDITION_1$	)	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	no	yes
$F)_1$	accolade_ouvrante	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	no	yes
FBLOCK_IF	else elif	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	yes	yes
BLOCK_IF	accolade_ouvrante	elif else ; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	no	yes
CASE	switch	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	no	yes
Fcase	id number boolean string call	accolade_fermante	no	no
FACTEUR	id number boolean string call	double_ampersand double_barre and or * mult slash div pourcentage mod modulo increment decrement + minus_sign comparator) virgule deux_points ;	no	no
$FFACTEUR_1$	deux_points	accolade_fermante	no	no
Fdeux_points	accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch (	accolade_fermante	no	no
FINSTRUCTIONS	case default	accolade_fermante	yes	no
BLOCK_CASE	case default	accolade_fermante	no	no
SHORTHAND	(	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	no	yes
CONDITION	(+ minus_sign id number boolean string call	double_ampersand double_barre and or) ;	no	no
INSTRUCTION	id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch (	; until accolade_fermante case default accolade_ouvrante id call return print printf scanf input log fprintf fscanf fread fwrite write read puts gets def function const let for do repeat while if switch ( ) deux_points	no	yes
VAR_DECLARATION	const let	; deux_points	no	no
TYPE	∅	∅	no	no
Fid_1	deux_points is	; deux_points	no	no
FVARS_TYPE	symbole_aff virgule	; deux_points	yes	no
$Fsymbole\_aff_1$	(+ minus_sign id number boolean string call	; deux_points	no	no
$FEXPRESSION_1$	virgule	; deux_points	yes	no
VARS"	id	; deux_points	no	no
VARS_TYPE	deux_points is	symbole_aff virgule ; deux_points	no	no
Fid_2	symbole_aff	; deux_points	no	no
Fsymbole_aff	(+ minus_sign id number boolean string call	; deux_points	no	no
EXPRESSION	(+ minus_sign id number boolean	double_ampersand double_barre and or	no	no

	string call	comparator) virgule deux_points ;		
FEXPRESSION	virgule	; deux_points	yes	no
IDS_CONST	id	; deux_points	no	no