put\_inptr\_first 定义一个操作，将输入指针放到行首

Program -->

program\_head program\_body {print: } }.

program\_head -->{print: main } program id{print: ( } ( identifier\_list){print: ) \n { } ;

identifier\_list —> id **{print: id.val }**temp1

program\_body --> declarations

subprogram\_declarations

compound\_statement

declarations --> VAR declaration ; | &

declaration --> identifier\_list :type { if(type.val != standrar\_type.val )

print: type.val put\_inptr\_first print: type.val2

else

put\_inptr\_first print: type.val

}temp0

type --> standard\_type {type.val = standard\_type .val}

| array [ digits .. digits ] of standard\_type

{type.val = [digits.val...digits.val] type.val2= standard\_type .val}

| record declaretion end **(此变量类型怎么表达？)**

standard\_type --> integer ｛standard\_type.val= int｝

| real ｛standard\_type.val= float｝

| Boolean ｛standard\_type.val= Boolean｝

| num .. num **(此变量类型怎么表达？)**

subprogram\_declarations —> temp2

subprogram\_declaration --> subprogram\_head{print: { }

declarations

compound\_statement{print: } }

subprogram\_head-->function id {print: id.val} arguments : standard\_type { put\_inptr\_first print: standard\_type.val } ; | procedure{print: void } id {print: id.val}arguments ;

arguments-->{print: ( }( parameter\_lists ) {print: ) } | &

parameter\_lists —> parameter\_list temp3

parameter\_list--> VAR identifier\_list: type

{ if(type.val != standrar\_type.val )

print: type.val put\_inptr\_first print: type.val2

else

put\_inptr\_first print: type.val }| identifier\_list : type

{ if(type.val != standrar\_type.val )

print: type.val put\_inptr\_first print: type.val2

else

put\_inptr\_first print: type.val }

compound\_statement--> begin

optional\_statements

end

optional\_statements--> statement\_list | &

statement\_list —> statement temp4

statement --> variable assignop {print: assignop.val }expression

| procedure\_call\_statement

| compound\_statement

| if {print: if }expression then {print: then }else {print: else } statement

| while {print: while } expression {print: do } statement

| read ( identifier\_list ) //怎么判断数据类型？

| write ( expr\_list )

variable--> id {print: id.val }| id {print: id.val } [ {print: [ } expression ] {print : ] }

procedure\_call\_statement--> id {print: id.val }

| id {print: id.val }({print: ( } expr\_list ) {print: ) }

expr\_list —> expression temp5

expression--> simple\_expr relop {print: relop.val } simple\_expr

| simple\_expr

simple\_expr —> term temp6 | sign term temp6

term-->factor temp7

factor --> id {factor.val = id.val}

| id {print : id.val } ( {print : ( } expr\_list ) {print: ) }

| id {print: id.val } [ {print: [ } expression ] {print : ] }

| num {factor.val = num.val}

| ( {print: ( } expression ) {print: ) }

| not factor {factor.val = ! factor.val1}

| true {factor.val = true}

| false {factor.val = false}

sign--> + {print : +}| - {print : - }

temp0—> {print: ; \n}declaration ;identifier\_list :type

{ if(type.val != standrar\_type.val )

print: type.val put\_inptr\_first print: type.val2

else

put\_inptr\_first print: type.val

}temp0 |&

temp1—> ,{print: ,} id {print: id.val}temp1 | &

temp2—> {print: \n}subprogram\_declaration ; temp2 | &

temp3-> ; {print: , }parameter\_list temp3 | &

temp4 —> ; {print: ; \n }statement temp4 | &

temp5 —> , {print: , }expression temp5 | & {print : ; \n }

temp6 —> addop {print: addop.val }term temp6 | & {print : ; \n }

temp7 -> mulop{print : mulop.val} factor temp7 |& {print : ; \n }