put\_inptr\_first 定义一个操作，将输入指针放到行首

put\_inptr\_end 将输入指针放到行尾

Program -->

program\_head program\_body {print: \nreturn 0; \n} }.

program\_head --> program {print: int main(int argc,char\* argv) \n { }

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 : {put\_ptr\_begin}type{put\_ptr\_end} temp0{print: \n}

type --> standard\_type

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

{put\_ptr\_end print: [digits.val...digits.val] }

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

standard\_type --> integer ｛print: int｝

| real ｛print: float｝

| Boolean ｛print: Boolean｝

| num .. num **(此变量类型怎么表达？)**

**////////////////////////////////**

subprogram\_declarations —> temp2

subprogram\_declaration --> subprogram\_head{ fun\_name=FileName print:\n{ \n}

declarations

compound\_statement{print:\n } \n}

subprogram\_head-->

function id {FileName.val=id.val} arguments:{put\_ptr\_first}{print: FileName.val}{put\_ptr\_end}{put\_ptr\_first}standard\_type{put\_ptr\_end};

|procedure id {temp.val=id.val}arguments{put\_ptr\_first}{print:temp.val}{put\_ptr\_end}{put\_ptr\_first}{print: void}{put\_ptr\_end} ;

arguments--> (parameter\_lists{put\_ptr\_first}{print:( }{put\_ptr\_end} {print: ) }) | &

parameter\_lists —> parameter\_list temp3

parameter\_list--> VAR identifier\_list: {put\_ptr\_begin}type{save\_type.val put\_ptr\_end print\_type} | identifier\_list : {put\_ptr\_begin} type {save\_type.val put\_ptr\_end print\_type}

compound\_statement--> begin

optional\_statements

end

optional\_statements--> statement\_list | &

statement\_list —> statement temp4

statement --> variable assignop {print: assignop.val (}expression {print:);\n} //reserve\_word[token[cur\_pointer].id]

| procedure\_call\_statement {print: ;\n}

| compound\_statement {print: ;\n}

| if {print: if (}expression{print:)\n} then {print: \n{\n } statement {print:}\n} else {print: else \n{\n} statement{print:\n}}

| while {print: while( } expression {print: )\n{ } statement{print:;\n}}

| read {read\_id\_num}( identifier\_list{function 141} )

| write {read\_temp5\_num } (expr\_list{function 142} )

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

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

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

expr\_list —> expression {fj++}temp5

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

| simple\_expr

simple\_expr —> term temp6 | sign term temp6

term-->factor temp7

factor --> id {print: id.val }

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

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

| num {print: num.val}

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

| not factor {put\_ptr\_first print: ! ( put\_ptr\_end print: )}

| true {print: true}

| false {print: false}

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

temp0—> ; {print: ; \n}identifier\_list : {put\_ptr\_begin}type temp0 |&

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

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

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

temp4 —> ; statement temp4 | &

temp5 —> , {print: }expression temp5 | &

temp6 —> addop {print: addop.val (}term temp6 {print: )}| &

temp7 -> mulop{print : mulop.val(} factor temp7 {print: )}|&