운영체제 과제
(BASIC Interpreter Source Code Analysis)

20191419 컴퓨터공학과 전유진

담당교수님 : 강영명교수님

구조체

```
struct node{
    int type;
/* system stack-> 1 for varie
 4 for begin 5 for end */

    char exp_data;
    int val;
    int line;
    struct node * next;

};

 typedef struct node Node;
```

1.node 구조체
1.1 정수 type;
1.2 문자 exp_data;
1.3 정수 val;
1.4 정수 line;
1.5 *next로 다음 node주소를 저장할 포인터

node구조체의 새로운 별칭을 Node이다

```
struct stack{
    Node * top;
};

typedef  struct stack Stack;
```

2. stack구조체
2.1 노드를 가리키는 top

stack구조체의 새로운 별칭을 Stack 이다

```
struct opnode{
    char op;
    struct opnode * next;
};
 typedef struct opnode opNode;
```

3. opnode 구조체
3.1 문자 op
3.2 *next로 다음opnode주소를 저장할 포인터

opnode구조체의 새로운 별칭을 opNode 이다.

```
struct opstack{
    opNode * top;
};

 typedef struct opstack OpStack;
```

4. opstack구조체
4.1 opNode를 가리키는 top

opstack 구조체의 새로운 별칭을 Opstack 이다

```
struct postfixnode{
    int val;
    struct postfixnode * next;
};

typedef struct postfixnode Postfixnode;
```

5 postfixnode 구조체
5.1 정수val ;
5.2 *next로 다음postfixnode주소를 저장할 포인터

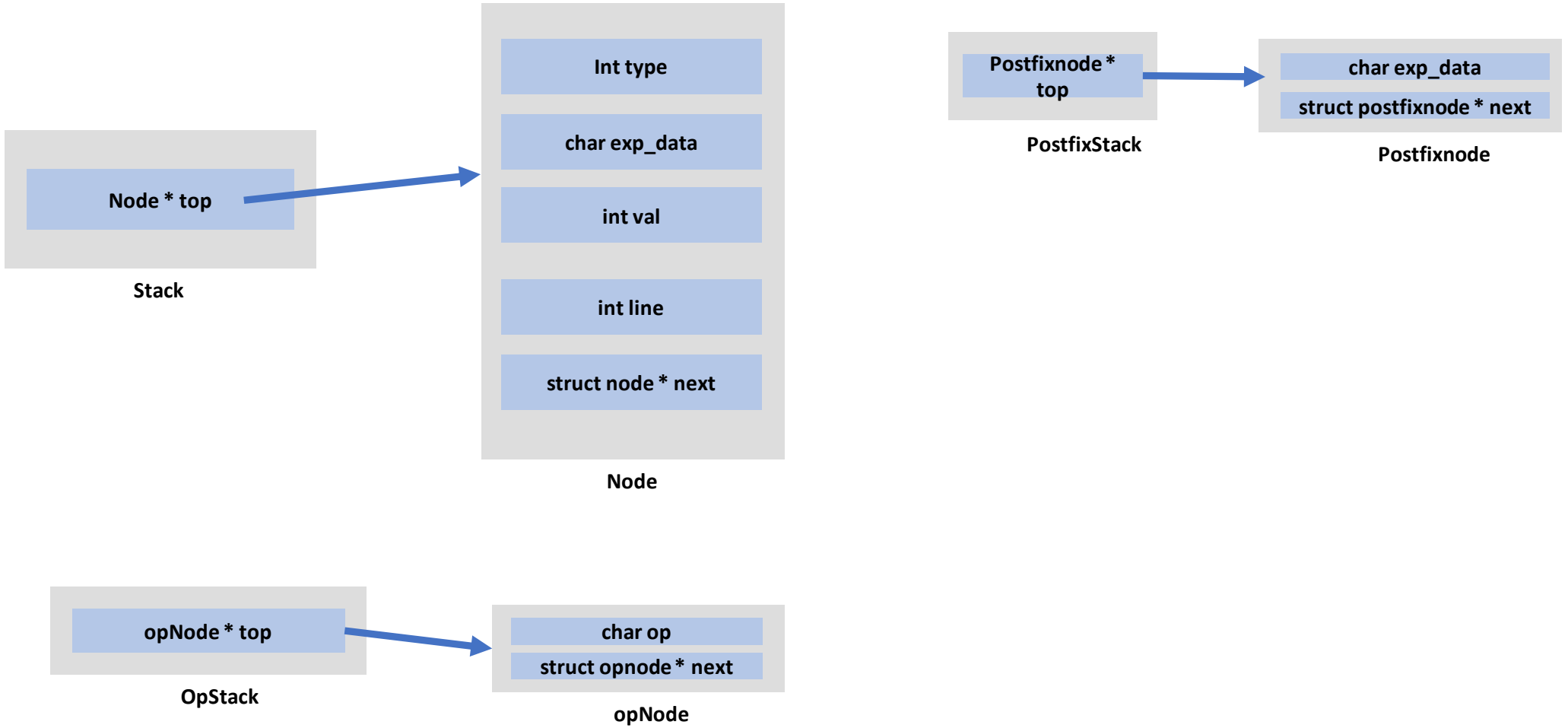
postfixnode구조체의 새로운 별칭을 Postfixnode 이다.

```
struct postfixstack{
    Postfixnode * top;
    };

 typedef struct postfixstack PostfixStack;
```

6. postfixstack구조체
6.1 Postfixnode를 가리키는 top

postfixstack 구조체의 새로운 별칭을 PostfixStack 이다

구조체 정리


Node * top
Stack
Int type
char exp_data
int val
int line
struct node * next
Node
Postfixnode * top
PostfixStack
char exp_data
struct postfixnode * next
Postfixnode
opNode * top
OpStack
char op
struct opnode * next
opNode

기능들

```
int GetVal(char,int *,Stack *);
int GetLastFunctionCall(Stack*);

Stack* FreeAll(Stack *);
```

정수자료형 GetVal 선언;
정수 자료형 GetLastFunctionCall선언
Stack구조체 자료형 FreeAll포인터 선언

push

```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```

Stack * Push(Node sNode,Stack *stck)

1. Node 자료형의 newnode포인터선언;
2. 만약 newnode는 Node타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   return NULL

3.아니라면
   newnode의 type은 sNode의 타입이 된다.
   newnode 의 val은 sNode의 val이된다.
   newnode 의 exp_data은 sNode의 exp_data이된다.
   newnode 의 line 은 sNode의 line이 된다.
   newnode 의 next 는 stck의 top이 된다.
   stck의 top은 newnode이다

반환값 stck

PushOp

```
OpStack * PushOp(char op,OpStack *opstck)
{
opNode *newnode;
if ((newnode=(opNode*)malloc(sizeof(opNode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->op=op;
newnode->next=opstck->top;
opstck->top=newnode;
return opstck;
}
}
```

OpStack * PushOp(char op,OpStack *opstck)

1. opNode자료형의 newnode포인터선언;
2. 만약 newnode는 opNode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   반환NULL
3. 아니라면,
   1. newnode의 op는 매개변수op이다.
   2. newnode의 next는 opstck의 top이다.
   3. opstck의 top은 newnode이다.

반환값 opstck

PushPostfix

```
PostfixStack * PushPostfix(int val,PostfixStack *poststck)
{
Postfixnode *newnode;
if ((newnode=(Postfixnode*)malloc(sizeof(Postfixnode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->val=val;
newnode->next=poststck->top;
poststck->top=newnode;
return poststck;
}
}
```

PostfixStack * PushPostfix(int val,PostfixStack *poststck)

1. Postfixnode구조체의 자료형인 newnode 포인터 선언
2. 만약 newnode는 Postfixnode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   NULL반환
3. 아니면라면,
   1. newnode의 val 은 매개변수 val이다
   2. newnode의 next 는 poststck의 top이다.
   3. poststck의 top 는 newnode이다.
   4. 반환값 poststck

PopOp

```
char PopOp(OpStack *opstck)
{
opNode *temp;
char op;
if (opstck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
op=opstck->top->op;
temp=opstck->top;
opstck->top=opstck->top->next;
free(temp);
return op;
}
return NULL;
}
```

char PopOp(OpStack *opstck)

1. opNode구조체의 자료형 temp 포인터 선언
2. 문자자료형 op선언
3. 만약 opstck의 top이 NULL일 경우
   printf("ERROR, empty stack..."); 출력
4. 아니라면,
   1. op는 opstck의 top의 op이다
   2. temp 는 opstck의 top이다.
   3. opstck의 top은 opstck의 top의 next이다.

   temp 동적메모리 해제

   반환값 op

반환값 NULL

PopPostfix

```
char PopPostfix(PostfixStack *poststck)
{
Postfixnode *temp;
int val;
if (poststck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
val=poststck->top->val;
temp=poststck->top;
poststck->top=poststck->top->next;
free(temp);
return val;
}
return NULL;
}
```

char PopPostfix(PstfixStack *poststck)

1. Postfixnode구조체의 자료형 temp 포인터 선언
2. 정수자료형인 val선언
3. 만약 poststck의 top이 NULL이라면
    1. ERROR, empty stack...console에 출력
4. 아니라면,
    1. val 은 poststck의 top의 val이다.
    2. tmep는 poststck의 top
    3. poststck의 top 은 poststck의 top의 next이다.
    4. temp의 동적메모리 해제
    5. 반환값 val

5.반환값 NULL

Pop

```
void Pop(Node * sNode,Stack *stck)
{
Node *temp;

if (stck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
sNode->exp_data=stck->top->exp_data;
sNode->type=stck->top->type;
sNode->line=stck->top->line;
sNode->val=stck->top->val;
temp=stck->top;
stck->top=stck->top->next;
free(temp);
}
}
```

void Pop(Node * sNode,Stack *stck)

1. Node구조체 자료형인 temp 포인터 선언
2. 만약 stck의 top 이 NULL이라면,
    1. ERROR, empty stack... console출력
3. 아니라면
    1. sNode의 exp_data 는 stck의 top의 exp_data이다.
    2. sNode의 type는 stck의 top의 type 이다
    3. sNode의 line는 stck의 top의 line이다
    4. sNode의 val는 stck의 top의 val이다
    5. temp는 stck의 top이다.
    6. stck의 top은 stck의 top의 next이다.
    7. temp동적메모리 해제

isStackEmpty

```
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)

1. 만약 stck의 top가 0이라면
    1. 반환값 1
2. 아니면 반환값 0

Priotry

```
int Priotry(char operator)
{
  if ((operator=='+') | (operator=='-'))
    return 1;
  else if ((operator=='/') | (operator=='*'))
    return 2;
return 0;
}
```

int Priotry(char operator)

1. 만약 operator 가 '+'이거나 operato가 '-'라면
    1. 반환값 1
2. 만약 operator가 '/'이거나 operator가 '*'이라면
    1. 반환값 2
3. 반환값 0

FreeAll

```
Stack * FreeAll(Stack * stck)
{
Node * temp;
Node * head;

  if (stck->top != NULL )
  {
    head=stck->top;
    do
    {

      temp=head;
      head=head->next;
      free(temp);

    } while (head->next!=NULL);
  }

return NULL;
}
```

Stack * FreeAll(Stack * stck)

1. Node구조체 자료형인 포인터 temp
2. Node구조체 자료형인 포인터 head
3. 만약 stck의 top이 NULL이 아니라면,
    1. head는 stck의top이다.
    2. do
        1. temp는 head이다.
        2. head는 head의 next이다.
        3. temp 동적 메모리 할당 해제한다.
    3. while (head의 next가 NULL이 아닐 때까지 반복)
4. 반환값 NULL

GetLastFunctionall

```
int GetLastFunctionCall(Stack *stck)
{
Node * head;

  if (stck->top == NULL )
  {
    printf("ERROR, empty stack...");
  }
  else
  {
    head=stck->top;
    do
    {
      if ( head->type==3 )
      {
        return head->line;
      }
      else
      {
        head=head->next;
      }
    } while (head->next!=NULL);
  }

  return 0;



}
```

int GetLastFunctionCall(Stack *stck)

1. Node의 구조체의 자료형인 포인터 head
2. 만약 stck의 top 이 NULL이라면,
    1. ERROR, empty stack... 이라고 콘솔 출력
3. 아니라면,
    1. head는 stck의 top이다.
    2. do
        1. head의 type은 3이라면
            1. 반환값 head의 line
        2. 아니라면
            1. head는 head의 next이다.
    3. while (head의 next가 NULL이 아니라면 반복)
4. 반환값 0

GetVal

```c
int GetVal(char exp_name,int * line,Stack *stck)
{

Node * head;
*line=0;
    if (stck->top == NULL )
    {
        printf("ERROR, empty stack...");
    }
    else
    {
        head=stck->top;
        do
        {
            if ( head->exp_data==exp_name )
            {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }

            }
            else
            {
                head=head->next;
            }
        } while (head->next!=NULL);
        /* check agin once more */
        if (head->exp_data==exp_name)
        {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }


        }
    }

    return -999;
}
```

int GetVal(char exp_name,int * line,Stack *stck)

1. 노드 구조체자료형인 head 포인트
2. *line은 0이다.
    1. 만약 stck의 top이 NULL일 경우
        1. ERROR, empty stack… 을 console 출력
    2. 아니면
        1. head는 stck의 top이다.
            1. **do**
                1. 만약 head의 exp_data 가 exp_name와 같다면
                    1. 만약 head의 type이 1이라면
                        1. 반환값 head의 val
                    2. 만약 head의 type이 2이라면
                        1. *line은 head의 line
                        2. 반환값 -1
                2. 만약 아니라면,
                    1. head는 head의 next
            2. **while(head의 next는 NULL이 아니라면 반복)**
            3. 만약 head의 exp_data 가 exp_name와 같다면
                1. 만약 head의 type이 1이라면
                    1. 반환값 head의 val
                2. 만약 head의 type이 2이라면
                    1. *line은 head의 line
                    2. 반환값 -1
    3. 반환값 -999

Main

```c
int main(int argc,char ** argv){
char line[4096];
    char dummy[4096];
    char lineyedek[4096];

    char postfix[4096];

    char * firstword;



//  int i;
    int val1;
    int val2;


    int LastExpReturn;
    int LastFunctionReturn=-999;
    int CalingFunctionArgVal;

    Node tempNode;


    OpStack * MathStack;

    //처리할 파일을 가리키고 있을 파일포인터
    FILE *filePtr;

    PostfixStack * CalcStack;

    int resultVal;

    Stack * STACK;

    int curLine=0;
    int foundMain=0;
        int WillBreak=0;


    MathStack->top=NULL;
    CalcStack->top=NULL;
    STACK->top=NULL;
    clrscr();
```

0. main함수의 매개변수

int argc는 메인함수에 전달되는 정보의 개수

char ** argv는 메인함수에 전달되는 실질적 정보로, 문자열 배열을 가리키는 포인터

1. line 문자열 배열크기가 4096인 선언
2. dummy문자열 배열크기가 4096인 선언
3. lineyedek 문자열 배열크기가 4096인 선언
4 . postfix문자열 배열크기가 4096인 선언
5. firstword 문자열 포인터 선언
6. val1 정수선언
7. val2 정수 선언
8. LastExpReturn; 정수 선언
9. 정수 LastFunctionRetur의 값은 -999이다
10. 정수선언 CalingFunctionArgVal;
11. OpStack구조체의 포인터 MathStack선언
12. FILE구조체에대한 filePtr 포인터 선언;
13. PostfixStack구조체의 CalcStack포인터 선언;
14. resultVal 정수선언;
15. Stack구조체의 * STACK 포인터 선언;
16. int curLine=0; //정수 curLine의 값은 0
17. int foundMain=0; //정수 foundMain 의 값은 0
18. int WillBreak=0; // 정수 WillBreak 의 값은 0
19. MathStack->top=NULL; // MathStack의 top은 NULL이다.
20. CalcStack->top=NULL; // CalcStack의 top은 NULL이다.
21. STACK->top=NULL; // STACK의 top은 NULL이다.
22. 화면을 지우는 기능

```c
//
if (argc!=2)
{
    /* if argument count is =1 */
    printf("Incorrect arguments!\n");
    printf("Usage: %s <inputfile.spl>",argv[0]);
    return 1;
}



/* open the file */
//메인함수에 전달되는 정보 inputfile1.spl 이 null
if ( ( filePtr=fopen(argv[1],"r") ) == NULL )
{
    printf("Can't open %s. Check the file please",argv[1]);
    return 2;
}
```

1. if문
만약 메인함수 전달되는 정보 개수가 2가 아니라면,
Incorrect arguments! 콘솔에 출력
Usage: 파일경로 <inputfile.spl> //argv[0]: 파일 경로
라고 console에 출력

return 1; //반환값 1

2. if문
if ((filePtr=fopen(argv[1],"r")) == NULL)
//메인함수에서 전달되는 실질적 정보 1번째꺼 읽기전용으로 파일 열어서 filePtr변수에 반환값 저장 만약 파일이 없다면 NULL반환
만약 파일이 없을 경우, Can't open argv[1]. Check the file please
을 consol에 출력하고 2를 반환

파일 읽기

```
//파일 스트림의 끝을 만날 때까지 반복
while (!feof(filePtr))
{

    int k=0;

    //입력 스트림에서 문자열 읽기

    fgets(line,4096, filePtr); /* read the file by Line by Line */
    /* scan for /t characters, get rid of them! */
    while(line[k]!='₩0')
    {
     if (line[k]=='₩t')
     {
        line[k]=' ';
     }

     k++;
    }

    strcpy(lineyedek,line);

    curLine++;
    tempNode.val=-999;
    tempNode.exp_data=' ';
    tempNode.line=-999;
    tempNode.type=-999;



    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }
    else{ ... }


}
```

1. 파일 스트림의 끝을 만날 때까지 반복
2. 정수 k의 값은 0
3. fget()
   fgets(파일 데이터를 저장할 변수, 읽어들일 최대 문자 수, 읽을 파일)

filePtr의파일을 최대 4095수까지 읽고 line배열에 저장.

4. whil문 line[k]이 null 이 아닐 시때까지 반복.
   4-1) 만약 line[k]가 \t(탭키)일 시, line[k]는 ‘ ‘으로 바꾼다.
   k증가

5. line을 lineyedek에 문자열 복사
6. curLine을 1증가
   tempNode.val= -999; // tempNode.val값을 -999로 대입
   tempNode.exp_data=' '; // tempNode.exp_data값을 ‘ ‘로 대입
   tempNode.line=-999;// tempNode.line 을 -999로 대입
   tempNode.type=-999;// tempNode.type 을 -999로 대입

7. 3가지로 나누어짐
   ① line이 begin일 경우
   ② line이 end일 경우
   ③ line이 begin과 end가 아닐경우

ⓐ begin

begin이라면

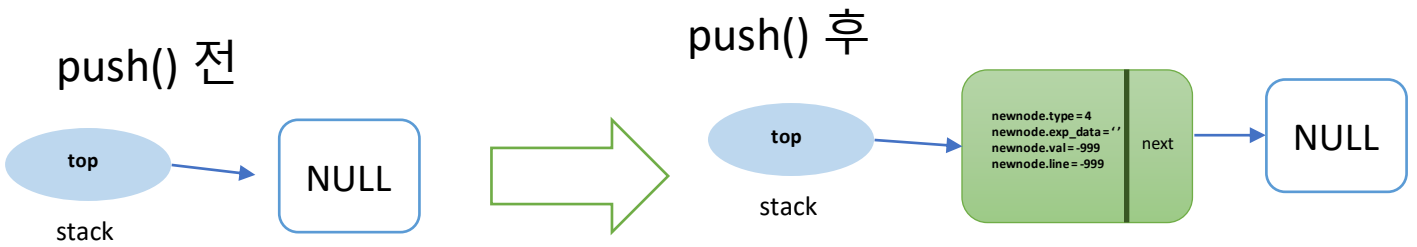
```
//begind이면
if (!strcmpi("begin\n",line) | !strcmpi("begin",line))
{
    //foundMain == 0이면 false 0이 아니면 True
    if (foundMain)
    {
        tempNode.type=4;
        STACK=Push(tempNode,STACK);
    }
}
```

Push

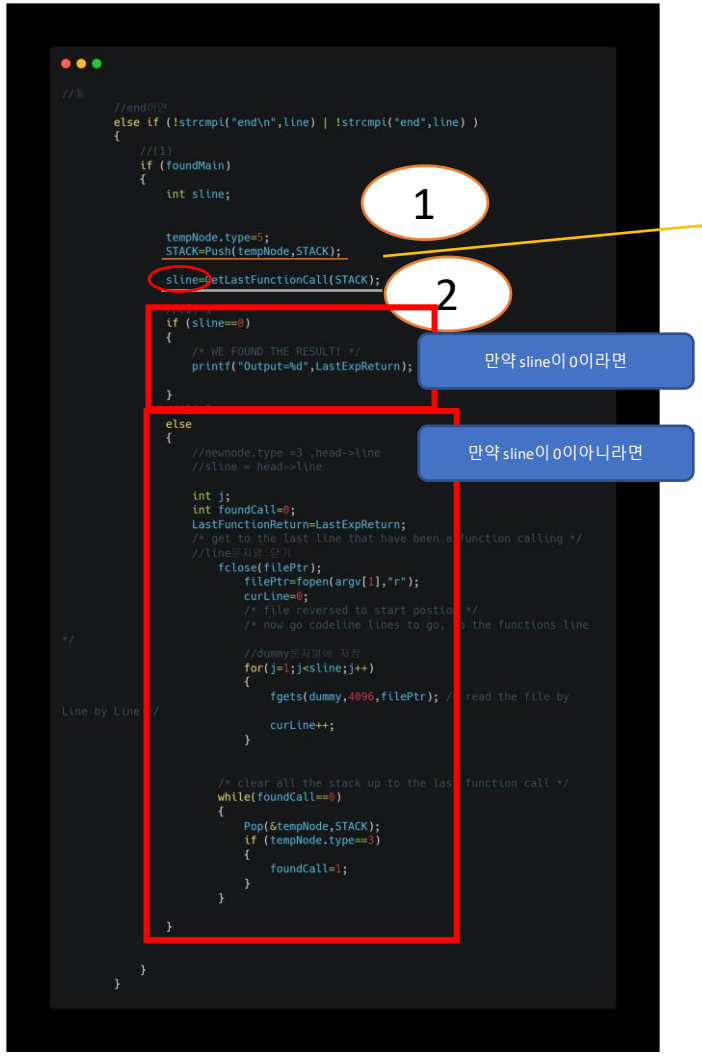
```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```

Node자료형의newnode포인터선언;
만약 newnode는 Node타입크기 만큼메모리를 할당받으며,
newnode가NULL일 경우 ERROR, Couldn't allocate memory... 출력
NULL리턴

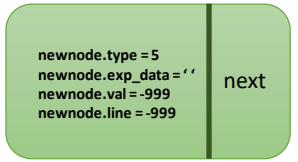
newnode.type = 4
newnode.val = -999
newnode.exp_data = ‘’
newnode.line = -999
newnode 의 next 는 stck의 top이 된다.
stck의 top은 newnode이다
반환값 stck



ⓑ line이 end일 때




```c
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```



Stack * Push(Node sNode,Stack *stck)

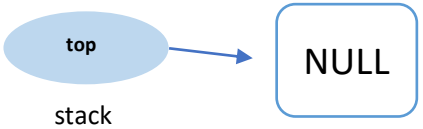
1. Node 자료형의 newnode포인터선언;
2. 만약 newnode는 Node타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   return NULL

3.아니라면
   newnode의 type은 5의 타입이 된다.
   newnode 의 val은 -999 이된다.
   newnode 의 line 은 -999 이 된다.
   newnode 의 next 는 stck의 top이 된다.
   stck의 top은 newnode이다

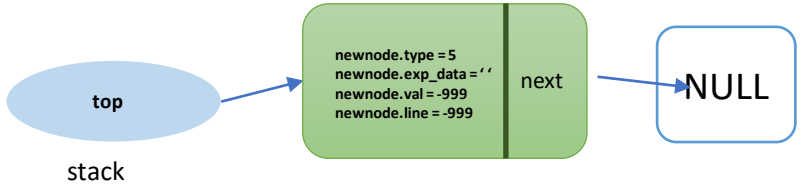
반환값 stck

1

push() 전



push() 후



ⓑ line이 end일 때

```
//ⓑ
//end이면
else if (!strcmpi("end\n",line) | !strcmpi("end",line) )
{
    //(1)
    if (foundMain)
    {
        int sline;


        tempNode.type=5;
        STACK=Push(tempNode,STACK);

        sline=GetLastFunctionCall(STACK);

        //(1)-1
        if (sline==0)
        {
            /* WE FOUND THE RESULT! */
            printf("Output=%d",LastExpReturn);

        }
        //(1)-2
        else
```

2

만약 sline이 0이라면,
Output= LastExpReturn의 값 을 콘솔 출력

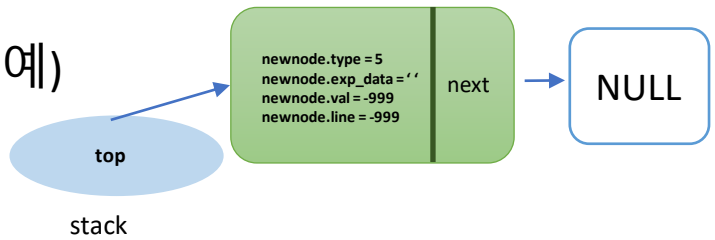
GetLastFunctionCall

```
int GetLastFunctionCall(Stack *stck)
{
Node * head;

  if (stck->top == NULL )
  {
    printf("ERROR, empty stack...");
  }
  else
  {
    head=stck->top;
    do
    {
      if ( head->type==3 )
      {
        return head->line;
      }
      else
      {
        head=head->next;
      }
    } while (head->next!=NULL);
  }

  return 0;



}
```

int GetLastFunctionCall(Stack *stck)

1. Node의 구조체의 자료형인 포인터 head
2. 만약 stck의 top 이 NULL이라면,
    1. ERROR, empty stack… 이라고 콘솔 출력
3. 아니라면,
    1. head는 stck의 top이다.
    2. do
        1. **head의 type은 3이라면**
            1. **반환값 head의 line**
        2. 아니라면
            1. head는 head의 next이다.
    3. while (head의 next가 NULL이 아니라면 반복)
4. **반환값 0**

예)



ⓑ line이 end일 때

만약 sline이 0이아니라면

```
//sline==0 이 아니라면
else
{
    //newnode.type =3 ,head->line
    //sline = head->line

    //정수 i선언
    int i;
    //정수 foundCall는 0
    int foundCall=0;
    //LastFunctionReturn은 LastExpReturn이다
    LastFunctionReturn=LastExpReturn;
    /* get to the last line that have been a function calling */
    //filePtr 닫기
        fclose(filePtr);
        /// filePtr= argv[1] 파일 읽기모드는 파일열기
            filePtr=fopen(argv[1],"r");

            //curLine은 0
            curLine=0;

            //1부터 sline보다 작을때까지 반복
            for(i=1;i<sline;i++)
            {
                //filePtr의파일을 최대 4095수까지 읽을 수 있으며, dummy배열에 저장.
                fgets(dummy,4096,filePtr); /* read the file by Line by Line */
                //curLine 1증가
                curLine++;
            }


        /* clear all the stack up to the last function call */
        //만일 foundCall이 0일 경우
        while(foundCall==0)
        {
            //Pop(&tempNode,STACK)실행
            Pop(&tempNode,STACK);
            //tempNode의 tupe이 3일 경우
            if (tempNode.type==3)
            {
                //foundCall은 1이다
                foundCall=1;
            }
        }
```

Pop

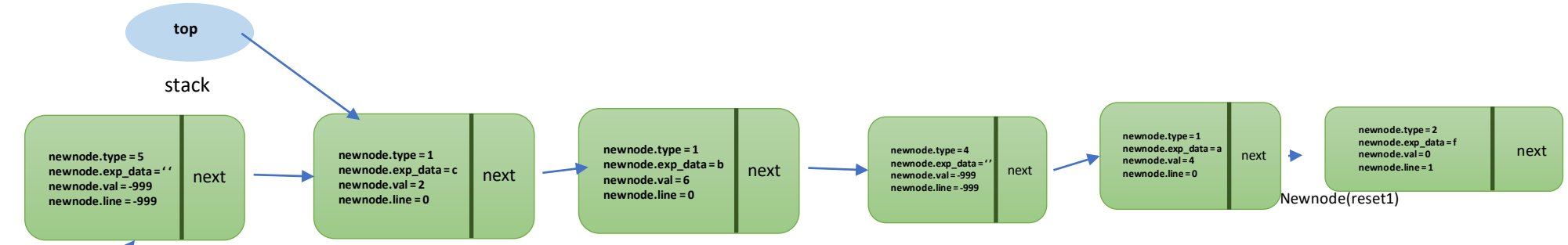
```
void Pop(Node * sNode,Stack *stck)
{
Node *temp;

if (stck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
sNode->exp_data=stck->top->exp_data;
sNode->type=stck->top->type;
sNode->line=stck->top->line;
sNode->val=stck->top->val;
temp=stck->top;
stck->top=stck->top->next;
free(temp);
}
}
```

void Pop(Node * sNode,Stack *stck)

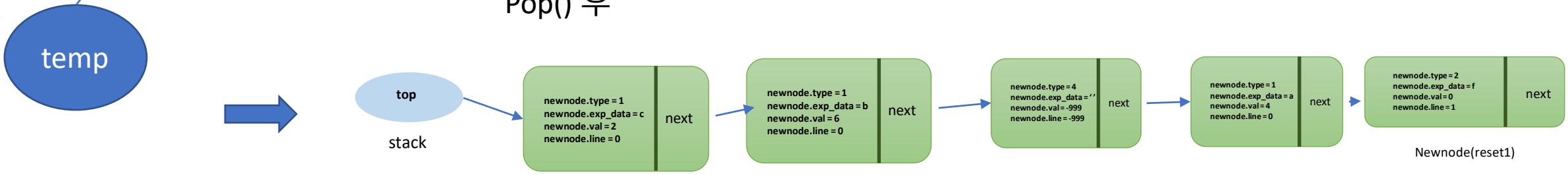
1. Node구조체 자료형인 temp 포인터 선언
2. 만약 stck의 top 이 NULL이라면,
    1. ERROR, empty stack... console출력
3. 아니라면
    1. sNode의 exp_data 는 stck의 top의 exp_data이다.
    2. sNode의 type는 stck의 top의 type 이다
    3. sNode의 line는 stck의 top의 line이다
    4. sNode의 val는 stck의 top의 val이다
    5. temp는 stck의 top이다.
    6. stck의 top은 stck의 top의 next이다.
    7. temp동적메모리 해제

예시)

Pop() 전


top
stack
newnode.type = 5
newnode.exp_data = ‘ ‘
newnode.val = -999
newnode.line = -999
next
newnode.type = 1
newnode.exp_data = c
newnode.val = 2
newnode.line = 0
next
newnode.type = 1
newnode.exp_data = b
newnode.val = 6
newnode.line = 0
next
newnode.type = 4
newnode.exp_data = ‘ ‘
newnode.val = -999
newnode.line = -999
next
newnode.type = 1
newnode.exp_data = a
newnode.val = 4
newnode.line = 0
next
newnode.type = 2
newnode.exp_data = f
newnode.val = 0
newnode.line = 1
next
Newnode(reset1)

Pop() 후


temp
top
stack
newnode.type = 1
newnode.exp_data = c
newnode.val = 2
newnode.line = 0
next
newnode.type = 1
newnode.exp_data = b
newnode.val = 6
newnode.line = 0
next
newnode.type = 4
newnode.exp_data = ‘ ‘
newnode.val = -999
newnode.line = -999
next
newnode.type = 1
newnode.exp_data = a
newnode.val = 4
newnode.line = 0
next
newnode.type = 2
newnode.exp_data = f
newnode.val = 0
newnode.line = 1
next
Newnode(reset1)

ⓒ begin과 end가 아니면

① firstword가 int라면

```
//ⓒ begin과 end가 아니면
else
{


   // 공백 -> 0으로 바꿈
   //line에서 공백 전의 첫문자
   firstword=strtok(line," ");
   //1.
   //int가 firstword이라면
   if (!strcmpi("int",firstword))
   {
      //(1)-1
      //foundMain ==0
      //foundMain == 0이면 false 0이 아니면 True
      if (foundMain)
      {
         //ttempNode.type 은 1
         tempNode.type=1; /*integer*/

         //자른 문자 다음부터 구분자 또 찾기
         firstword=strtok(NULL," ");

         //tempNode.exp_data의 exp_data = firstword[0]
         tempNode.exp_data=firstword[0];


         //자른 문자 다음부터 구분자  찾기
         firstword=strtok(NULL," ");

         //(2)-1
         /* check for = */
         //firstword가 '='이라면 true
         if (!strcmpi("=",firstword))
         {
            //firstword는 자른 문자 다음부터 구분자  찾기은 값이다.
            firstword=strtok(NULL," ");
         }

         //문자열을 정수 타입 변화 ex) 1
         //tempNode의 val = firstword를 정수 변환
         tempNode.val=atoi(firstword);
         //tempNode의 line은  0
         tempNode.line=0;
         //STACK은 Push(tempNode,STACK)한 값
         STACK=Push(tempNode,STACK);
      }
   }
```

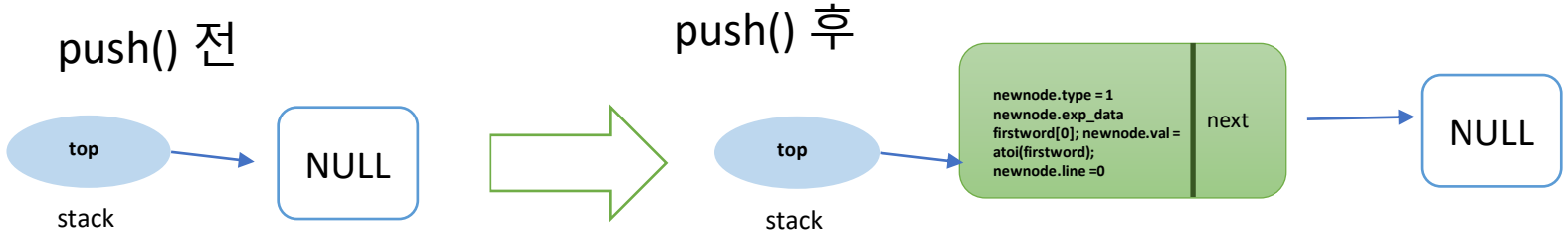
```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```

Stack * Push(Node sNode,Stack *stck)

1. Node 자료형의 newnode포인터선언;
2. 만약 newnode는 Node타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory… 출력
   return NULL

3.아니라면
   newnode의 type은 sNode의 타입이 된다.
   newnode 의 val은 sNode의 val이된다.
   newnode 의 exp_data은 sNode의 exp_data이된다.
   newnode 의 line 은 sNode의 line이 된다.
   newnode 의 next 는 stck의 top이 된다.
   stck의 top은 newnode이다

반환값 stck



ⓒ begin과 end가 아니면

② firstword가 function이라면

```
//function가 firstword이라면
else if (!strcmpi("function",firstword))
{
    //tempNode.type는 2이다
    tempNode.type=2;

    //  //자른 문자 다음부터 구분자  찾기
    firstword=strtok(NULL," ");

    //tempNode.exp_data는 firstword[0]이다
    tempNode.exp_data=firstword[0];

    //tempNode.line 는 curLine이다
    tempNode.line=curLine;
    //tempNode.val은 0이다.
    tempNode.val=0;
    //STACK은 Push(tempNode,STACK); 에서의 반환값이다.
    STACK=Push(tempNode,STACK);
```

```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```

Stack * Push(Node sNode,Stack *stck)

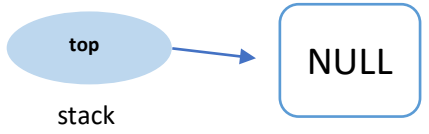
1. Node 자료형의 newnode포인터선언;
2. 만약 newnode는 Node타입크기 만큼메모리를 할당받으며, newnode가 NU
   ERROR, Couldn't allocate memory... 출력
   return NULL

3.아니라면

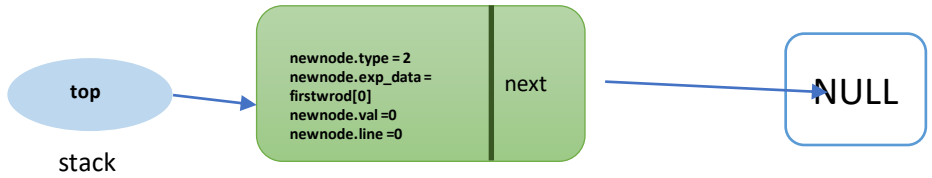
newnode의 type은 sNode의 타입이 된다.
newnode 의 val은 sNode의 val이된다.
newnode 의 exp_data은 sNode의 exp_data이된다.
newnode 의 line 은 sNode의 line이 된다.
newnode 의 next 는 stck의 top이 된다.
stck의 top은 newnode이다

반환값 stck

push() 전



push() 후



ⓒ begin과 end가 아니면

② firstword가 function이라면

```
STACK=Push(tempNode,STACK);

//만약 firstword배열의 0번째가 'm'이며 firstword배열의 1번째가 'a'이며 firstword배열의 2번째가 'i'이며 firstword배열의 3번째까 'n'이라면
if ( (firstword[0]=='m') & (firstword[1]=='a') & (firstword[2]=='i') & (firstword[3]=='n') )
{
    /*printf("Found function main() in line %d. Start
    //foundMain 은 1이다.
    foundMain=1;


}
//만약 firstword배열의 0번째가 'm'이며 firstword배열의 1번째가 'a'이며 firstword배열의 2번째가 'i'이며 firstword배열의 3번째까 'n'이 아니라면
else
{
    //(3)
    //foundMain == 0이면 false 0이 아니면 True
    if (foundMain)
    {
        //" "공백 기준으로 다음
        ////    //자른 문자 다음부터 구분자  찾기
        firstword=strtok(NULL," ");
        //tempNode.type 은 1이다
        tempNode.type=1;
        //tempNode.exp_data는 firstword[0]이다
        tempNode.exp_data=firstword[0];
        //tempNode.val는 CalingFunctionArgVal
        tempNode.val=CalingFunctionArgVal;
        //tempNode의 line은 0이다;
        tempNode.line=0;
        ////STACK은 Push(tempNode,STACK); 에서의 반환값이다.
        STACK = Push(tempNode, STACK);

    }

}
```

만약 firstword배열의 0번째가 'm'이며 firstword배열의 1번째가 'a'이며 firstword배열의 2번째가 'i'이며 firstword배열의 3번째까 'n'이라면

1

만약 firstword배열의 0번째가 'm'이며 firstword배열의 1번째가 'a'이며 firstword배열의 2번째가 'i'이며 firstword배열의 3번째까 'n ' 이 아니라면

2

2

```
//(3)
//foundMain == 0
if (foundMain)
{
    ////    //자른 문자 다음부터 구분자  찾기
    firstword=strtok(NULL," ");
    //tempNode.type 은 1이다
    tempNode.type=1;
    //tempNode.exp_data는 firstword[0]이다
    tempNode.exp_data=firstword[0];
    //tempNode.val는 CalingFunctionArgVal
    tempNode.val=CalingFunctionArgVal;
    //tempNode의 line은 0이다;
    tempNode.line=0;
    ////STACK은 Push(tempNode,STACK); 에서의 반환값이다.
    STACK = Push(tempNode, STACK);
}
```

foundMain == 0이면 False 0이 아니면 True

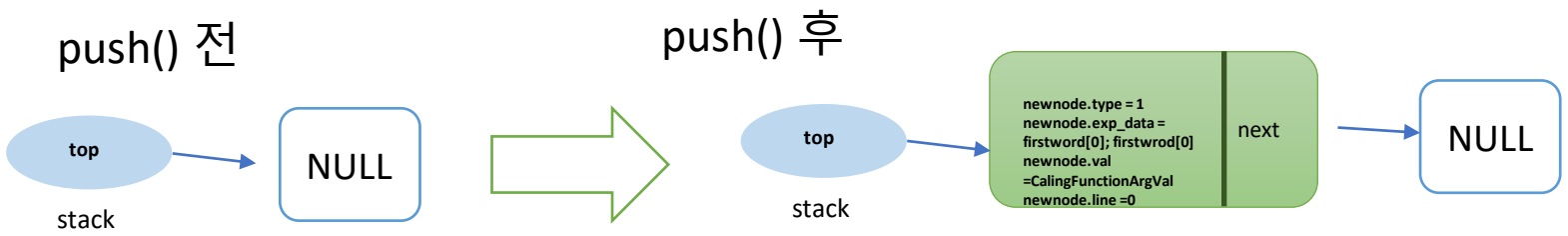
```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```

Stack * Push(Node sNode,Stack *stck)

1. Node 자료형의 newnode포인터선언;
2. 만약 newnode는 Node타입크기 만큼메모리를 할당받으며, newnode
   ERROR, Couldn't allocate memory... 출력
   return NULL

3.아니라면
   newnode의 type은 sNode의 타입이 된다.
   newnode 의 val은 sNode의 val이된다.
   newnode 의 exp_data은 sNode의 exp_data이된다.
   newnode 의 line 은 sNode의 line이 된다.
   newnode 의 next 는 stck의 top이 된다.
   stck의 top은 newnode이다

반환값 stck



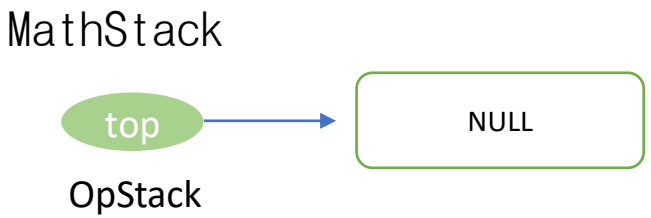
ⓒ begin과 end가 아니면

③ lineyedek[0]이 '('이라면

```
//firstword[0]이 (이라면
else if (firstword[0]=='(')
{

//foundMain == 0이면 False 0이 foundMain == 0이 아니면 True
//foundMain이 1이면 실행
if (foundMain)
    {

    //정수 i는 0
    int i=0;
    //정수 y는 0
    int y=0;

    //OpStack * MathStack
    //MathStack.top은 NULL
    MathStack->top=NULL;
    /* now make the postfix calculcation */
```

foundMain == 0이면 False 0이 아니면 True

이 PC에 저장됨



ⓒ begin과 end가 아니면

③ lineyedek[0]이 '('이라면

```
//lineyedek[i]이 NULL이 아닐때까지 반복
while(lineyedek[i]!='₩x0')
{
    /* evulate the function */
    //lineyedek[i]가 숫자라면 True
    if (isdigit(lineyedek[i])){ ... }
     /* else if (lineyedek[i]=='(')
    {
       MathStack=PushOp(lineyedek[i],MathStack);
    }*/

    //lineyedek[i]이 ')'이라면
    else if (lineyedek[i]==')'){ ... }

    ////(3)-3-3
    //lineyedek[i]이 '+'이거나 lineyedek[i]이 '-' 이거나 lineyedek[i]이 '+' 이거나 lineyedek[i]이 '/'이라면
    else if ((lineyedek[i]=='+') | (lineyedek[i]=='-') | (lineyedek[i]=='*') | (lineyedek[i]=='/')){ ... }

    //알파벳 대문자 "A-Z"는 1을 반환.알파벳 소문자 'a-z"는 2를 반환.
    //lineyedek[i]가 영어라면
    else if (isalpha(lineyedek[i])>0){ ... }

    //1 증가
    i++;
}//while문
```

1

2

3

4

ⓒ begin과 end가 아니면

③ lineyedek[0]이 '('이라면

③-1 //lineyedek[i]가 숫자라면 True

1

```
/* evulate the function */
//lineyedek[i]가 숫자라면 True
if (isdigit(lineyedek[i])) {
    //postfix[y에 lineyedek[i] 넣기
    postfix[y]=lineyedek[i];

    y++;  //y값 1증가

}
```

예시)

postfix[0] = 6

ⓒ begin과 end가 아니면

③ lineyedek[0]이 '('이라면

③-2 lineyedek[i]이 ')'이라면

2

```
//lineyedek[i]이 ')'이라면
else if (lineyedek[i]==')')
{
    //(3)-3-2-1
    ////00이면 true 10이면 false
    //0일때
    if (!isStackEmpty(MathStack) != 0 )
    {
        //postfix[y]에 PopOp(MathStack)의 반환값 넣기
        postfix[y]=PopOp(MathStack);
        y++; // 1증가
    }
}
```

ⓐ

ⓑ

postfix[1] = +

isStackEmpty

```
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)

1. 만약 stck의 top가 0이라면
   1. 반환값 1
2. 아니면 반환값 0

PopOp

```
char PopOp(OpStack *opstck)
{
opNode *temp;
char op;
if (opstck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
op=opstck->top->op;
temp=opstck->top;
opstck->top=opstck->top->next;
free(temp);
return op;
}
return NULL;
}
```

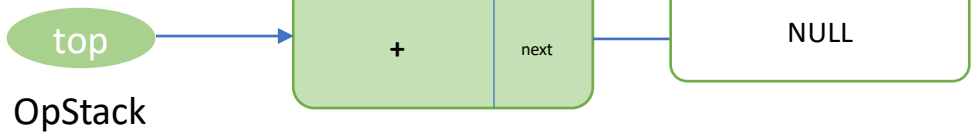
char PopOp(OpStack *opstck)

1. opNode구조체의 자료형 temp 포인터 선언
2. 문자자료형 op선언
3. 만약 opstck의 top이 NULL일 경우
   printf("ERROR, empty stack..."); 출력
4. 아니라면,
   1. op는 opstck의 top의 op이다
   2. temp 는 opstck의 top이다.
   3. opstck의 top은 opstck의 top의 next이다.
   temp 동적메모리 해제
   반환값 op

반환값 NULL

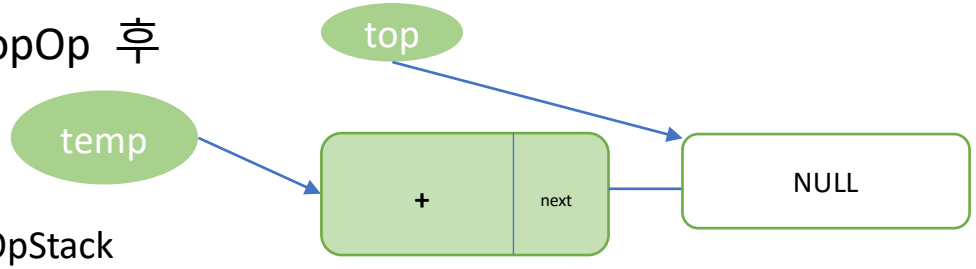
전

PopOp 전



OpStack

PopOp 후



OpStack

ⓒ begin과 end가 아니면

③ lineyedek[0]이 '('이라면

③-3 lineyedek[i]이 '+'이거나 lineyedek[i]이 '-' 이거나 lineyedek[i]이 '+' 이거나 lineyedek[i]이 '/'이라면

3

```
//lineyedek[i]이 '+'이거나 lineyedek[i]이 '-' 이거나 lineyedek[i]이 '+' 이거나 lineyedek[i]이 '/'이라면
else if ((lineyedek[i]=='+') | (lineyedek[i]=='-') | (lineyedek[i]=='*') | (lineyedek[i]=='/'))
{
    ////(3)-3-3-1
    /*operators*/
    //00이면 false 10이면 true
    //isStackEmpty(MathStack)이 00이 아니라면
    if (isStackEmpty(MathStack) != 0 )
    {
        // MathStack은 PushOp(lineyedek[i],MathStack)의 반환값
        MathStack=PushOp(lineyedek[i],MathStack);
    }

    //+)+(+
    //isStackEmpty(MathStack)이 00이라면
    else
    {
        //Priotry(MathStack->top->op)이 Priotry(lineyedek[i])의 반환값보다 크거나 같을때
        if (Priotry(lineyedek[i]) <= Priotry(MathStack->top->op) )
        {
            //postfix[y]은 PopOp(MathStack)반환값
            postfix[y]=PopOp(MathStack);
            //y값 1증가
            y++;

            //MathStack은 PushOp(lineyedek[i],MathStack)이다
            MathStack=PushOp(lineyedek[i],MathStack);


        }
        //Priotry(MathStack->top->op)이 Priotry(lineyedek[i])이 작을 때
        else
        {
            //MathStack은 PushOp(lineyedek[i],MathStack)의 반환값
            MathStack=PushOp(lineyedek[i],MathStack);
        }

    }
}
```

isStackEmpty(MathStack)의 반환값이 00이 아니라면

isStackEmpty(MathStack)의 반환값이 00이라면

isStackEmpty(MathStack)의 반환값이 0이 아니라면

ⓐ

```
////(3)-3-3
//lineyedek[i]이 '+'이거나 lineyedek[i]이 '-' 이거나 lineyedek[i]이 '+' 이거나 lineyedek[i]이 '/'이라면
else if ((lineyedek[i]=='+') | (lineyedek[i]=='-') | (lineyedek[i]=='*') | (lineyedek[i]=='/'))
{
    ////(3)-3-3-1
    /*operators*/
    //00이면 false 10이면 true
    //처음일 때
    if (isStackEmpty(MathStack) != 0 ) //1일때 - stck->top==0
    {
/* if stack empty push the operator to stack */
      //+
      MathStack=PushOp(lineyedek[i],MathStack);
    }
    ////(3)-3-3-2

    //0일때 - stck -top = 00이 아닐때
    //처음이 아닐때
    else { ... }
}
```

ⓑ

isStackEmpty

```
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)

1. 만약 stck의 top가 0이라면
   1. 반환값 1
2. 아니면 반환값 0

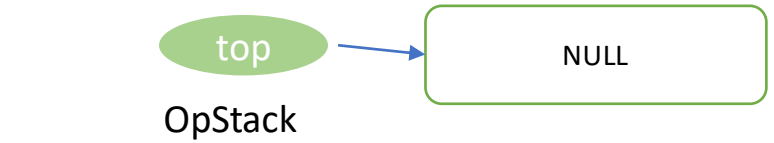
PushOp

```
OpStack * PushOp(char op,OpStack *opstck)
{
opNode *newnode;
if ((newnode=(opNode*)malloc(sizeof(opNode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->op=op;
newnode->next=opstck->top;
opstck->top=newnode;
return opstck;
}
}
```

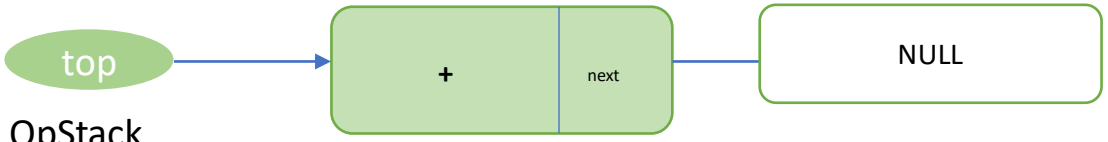
OpStack * PushOp(char op,OpStack *opstck)

1. opNode자료형의 newnode포인터선언;
2. 만약 newnode는 opNode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   반환NULL
3. 아니라면,
   1. newnode의 op는 매개변수+이다.
   2. newnode의 next는 opstck의 top이다.
   3. opstck의 top은 newnode이다.

반환값 opstck



PushOp



isStackEmpty(MathStack)의 반환값이 00이 아니라면

```
//isStackEmpty(MathStack)의 반환값이 00이라면
else
{
    //Priotry(MathStack->top->op0| Priotry(lineyedek[i])의 반환값보다 크거나 같을때
    if (Priotry(lineyedek[i]) <= Priotry(MathStack->top->op) )
    {
        //postfix[y]은 PopOp(MathStack)반환값
        postfix[y]=PopOp(MathStack);

        //y값 1증가
        y++;

        //MathStack은 PushOp(lineyedek[i],MathStack)이다
        MathStack=PushOp(lineyedek[i],MathStack);


    }
    //Priotry(MathStack->top->op0| Priotry(lineyedek[i])이 작을 때
    else
    {
        //MathStack은 PushOp(lineyedek[i],MathStack)의 반환값
        MathStack=PushOp(lineyedek[i],MathStack);
    }

}
}
```

1

Priotry(MathStack->top->op)의 반환값이
Priotry(lineyedek[i])의 반환값보다 크거나 같을때

2

Priotry(lineyedek[i])의 반환값이 Priotry(MathStack->top->op)의 반환값보다 클 때

1

Priotry(MathStack->top->op)의 반환값이
Priotry(lineyedek[i])의 반환값보다 크거나 같을때

```
//*)
//isStackEmpty(MathStack)이 OO이라면
else
{
    //Priotry(MathStack->top->op)이 Priotry(lineyedek[i])의 반환값보다 크거나 같을때
    if (Priotry(lineyedek[i]) <= Priotry(MathStack->top->op) )
    {
        //postfix[y]은 PopOp(MathStack)반환값
        postfix[y]=PopOp(MathStack);
        //y값 1증가
        y++;

        //MathStack은 PushOp(lineyedek[i],MathStack)이다
        MathStack=PushOp(lineyedek[i],MathStack);

    }
    //Priotry(MathStack->top->op)이 Priotry(lineyedek[i])이 작을 때
    else
    {
        //MathStack은 PushOp(lineyedek[i],MathStack)의 반환값
        MathStack=PushOp(lineyedek[i],MathStack);
    }

}
```

if(1 <= 2)

Priotry

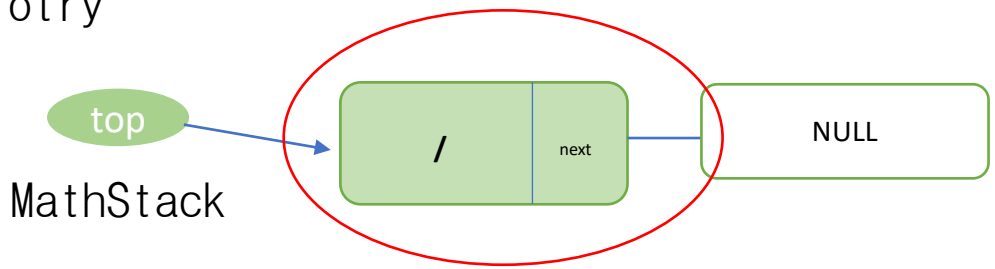
```
int Priotry(char operator)
{
  if ((operator=='+') | (operator=='-'))
    return 1;
  else if ((operator=='/') | (operator=='*'))
    return 2;
return 0;
}
```

int Priotry(char operator)

1. 만약 operator 가 '+'이거나 operato가 '-'라면
   1. 반환값 1
2. 만약 operator가 '/'이거나 operator가 '*'이라면
   1. 반환값 2
3. 반환값 0

예)

Priotry



예) Priotry (lineyedek[i])
lineyedek[i] == '+' 일 때, **반환값 1**

Priotry (MathStack->top->op])은 /일때, **반환값 2**

1

Priotry(MathStack->top->op)의 반환값이
Priotry(lineyedek[i])의 반환값보다 크거나 같을때

```
//*)*(*
//isStackEmpty(MathStack)이 OO이라면
else
{
    //Priotry(MathStack->top->op이 Priotry(lineyedek[i])의 반환값보다 크거나 같을때
    if (Priotry(lineyedek[i]) <= Priotry(MathStack->top->op) )
    {
        //postfix[y]은 PopOp(MathStack)반환값
        postfix[y]=PopOp(MathStack);
        //y값 1증가
        y++;

        //MathStack은 PushOp(lineyedek[i],MathStack)이다
        MathStack=PushOp(lineyedek[i],MathStack);


    }
    //Priotry(MathStack->top->op이 Priotry(lineyedek[i])이 작을 때
    else
    {
        //MathStack은 PushOp(lineyedek[i],MathStack)의 반환값
        MathStack=PushOp(lineyedek[i],MathStack);
    }

}
```

ⓐ

PopOp

```
char PopOp(OpStack *opstck)
{
opNode *temp;
char op;
if (opstck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
op=opstck->top->op;
temp=opstck->top;
opstck->top=opstck->top->next;
free(temp);
return op;
}
return NULL;
}
```

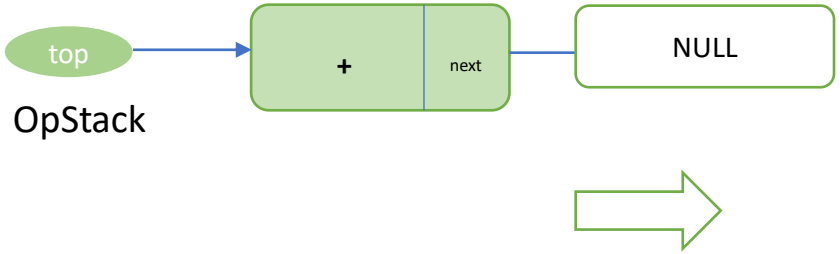
char PopOp(OpStack *opstck)

1. opNode구조체의 자료형 temp 포인터 선언
2. 문자자료형 op선언
3. 만약 opstck의 top이 NULL일 경우
   printf("ERROR, empty stack..."); 출력
4. 아니라면,
   1. op는 opstck의 top의 op이다
   2. temp 는 opstck의 top이다.
   3. opstck의 top은 opstck의 top의 next이다.
   temp 동적메모리 해제
   반환값 op

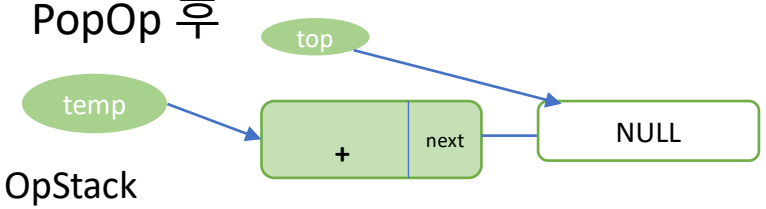
반환값 NULL

예시)

전



PopOp 후



1

Priotry(MathStack->top->op)의 반환값이
Priotry(lineyedek[i])의 반환값보다 크거나 같을때

```
//*)*(*
//isStackEmpty(MathStack)이 아니라면
else
{
    //Priotry(MathStack->top->op이 Priotry(lineyedek[i])의 반환값보다 크거나 같을때
    if (Priotry(lineyedek[i]) <= Priotry(MathStack->top->op) )
    {
        //postfix[y]은 PopOp(MathStack)반환값
        postfix[y]=PopOp(MathStack);
        //y값 1증가
        y++;

        //MathStack은 PushOp(lineyedek[i],MathStack)이다
        MathStack=PushOp(lineyedek[i],MathStack);

    }
    //Priotry(MathStack->top->op이 Priotry(lineyedek[i])이 작을 때
    else
    {
        //MathStack은 PushOp(lineyedek[i],MathStack)의 반환값
        MathStack=PushOp(lineyedek[i],MathStack);
    }

}
```

PushOp

```
OpStack * PushOp(char op,OpStack *opstck)
{
opNode *newnode;
if ((newnode=(opNode*)malloc(sizeof(opNode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->op=op;
newnode->next=opstck->top;
opstck->top=newnode;
return opstck;
}
}
```

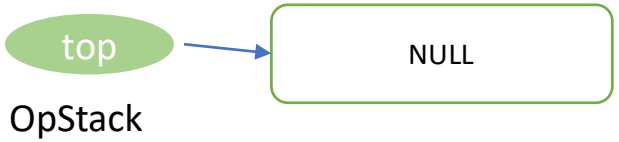
OpStack * PushOp(char op,OpStack *opstck)

1. opNode자료형의 newnode포인터선언;
2. 만약 newnode는 opNode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   반환NULL
3. 아니라면,
   1. newnode의 op는 매개변수+이다.
   2. newnode의 next는 opstck의 top이다.
   3. opstck의 top은 newnode이다.

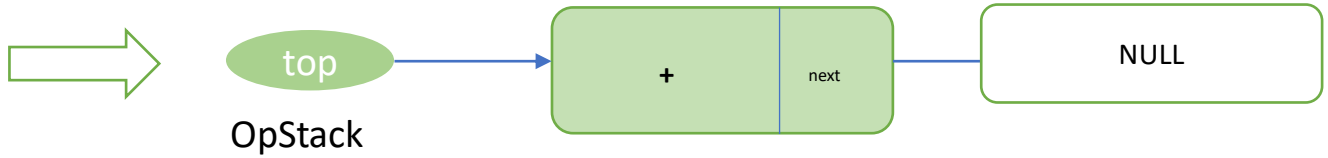
반환값 opstck

예시)

PushOp 전



PushOp 후



Priotry(lineyedek[i])의 반환값이 Priotry(MathStack->top->op)의 반환값보다 클 때

```
else
{
    //Priotry(MathStack->top->op이 Priotry(lineyedek[i])의 반환값보다 크거나 같을때
    if (Priotry(lineyedek[i]) <= Priotry(MathStack->top->op) )
    {
        //postfix[y]은 PopOp(MathStack)반환값
        postfix[y]=PopOp(MathStack);
        //y값 1증가
        y++;

        //MathStack은 PushOp(lineyedek[i],MathStack)이다
        MathStack=PushOp(lineyedek[i],MathStack);


    }
    //Priotry(MathStack->top->op이 Priotry(lineyedek[i])이 작을 때
    else
    {
        //MathStack은 PushOp(lineyedek[i],MathStack)의 반환값
        MathStack=PushOp(lineyedek[i],MathStack);
    }

}
```

Priotry

```
int Priotry(char operator)
{
  if ((operator=='+') | (operator=='-'))
    return 1;
  else if ((operator=='/') | (operator=='*'))
    return 2;
return 0;
}
```

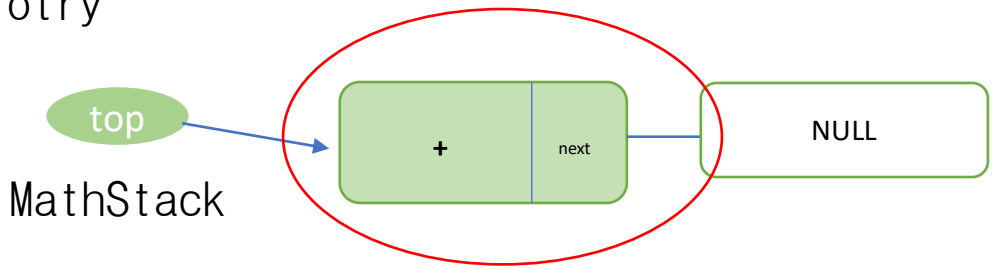
int Priotry(char operator)

1. 만약 operator 가 '+'이거나 operato가 '-'라면
   1. 반환값 1
2. 만약 operator가 '/'이거나 operator가 '*'이라면
   1. 반환값 2
3. 반환값 0

if(2 >1)

예)

Priotry



예) Priotry (lineyedek[i])
lineyedek[i] == '/' 일 때, **반환값 2**

Priotry (MathStack->top->op])은 +일때, **반환값 1**

Priotry(lineyedek[i])의 반환값이 Priotry(MathStack->top->op)의 반환값보다 클 때

```
//*)*(*
//isSta
else
{
    //Priotry(MathStack->top->op이 Priotry(lineyedek[i])의 반환값보다 크거나 같을때
    if (Priotry(lineyedek[i]) <= Priotry(MathStack->top->op) )
    {
        //postfix[y]은 PopOp(MathStack)반환값
        postfix[y]=PopOp(MathStack);
        //y값 1증가
        y++;

        //MathStack은 PushOp(lineyedek[i],MathStack)이다
        MathStack=PushOp(lineyedek[i],MathStack);


    }
    //Priotry(MathStack->top->op이 Priotry(lineyedek[i])이 작을 때
    else
    {
        //MathStack은 PushOp(lineyedek[i],MathStack)의 반환값
        MathStack=PushOp(lineyedek[i],MathStack);
    }

}
```

PushOp

```
OpStack * PushOp(char op,OpStack *opstck)
{
opNode *newnode;
if ((newnode=(opNode*)malloc(sizeof(opNode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->op=op;
newnode->next=opstck->top;
opstck->top=newnode;
return opstck;
}
}
```

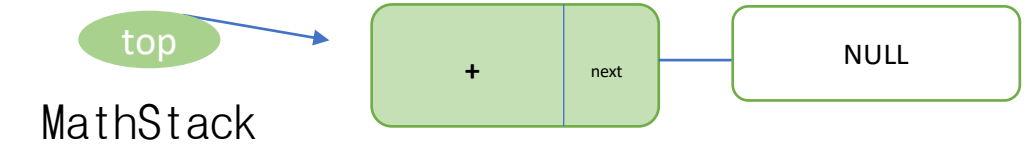
OpStack * PushOp(char op,OpStack *opstck)

1. opNode자료형의 newnode포인터선언;
2. 만약 newnode는 opNode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   반환NULL
3. 아니라면,
   1. newnode의 op는 매개변수+이다.
   2. newnode의 next는 opstck의 top이다.
   3. opstck의 top은 newnode이다.

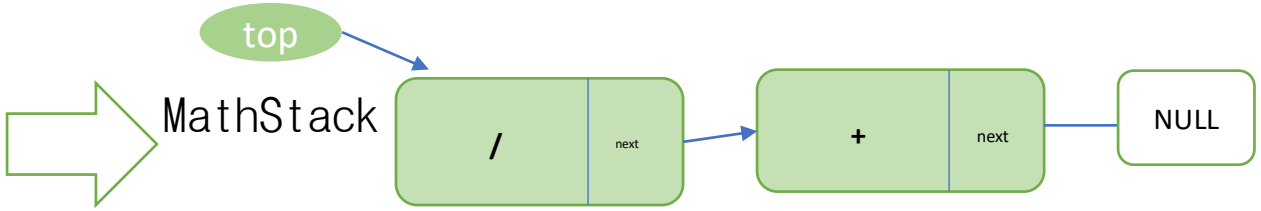
반환값 opstck

예시)

PushOp 전



PushOp 후



ⓒ begin과 end가 아니면

③ lineyedek[0]이 '('이라면

③-4 lineyedek[i]가 영어라면

4

```
//알파벳 대문자 "A-Z"는 1을 반환,알파벳 소문자 'a-z"는 2를 반환.
//lineyedek[i] 영어라면
else if (isalpha(lineyedek[i])>0)
{
    //정수 codeline는 0이다
    //정수 dummyint는 0이다
    int codeline=0;
    int dummyint=0;
    /*look if it's a variable or function call                    */
    int retVal=0;
    //retVal은 GetVal(lineyedek[i],&codeline,STACK);의 반환값
    retVal=GetVal(lineyedek[i],&codeline,STACK);
    //int codeline =1


    //만일 retVal이 -1이 아니면서 -9999이 아니라면
```

GetVal

```
int GetVal(char exp_name,int * line,Stack *stck)
{
Node * head;
*line=0;
    if (stck->top == NULL )
    {
        printf("ERROR, empty stack...");
    }
    else
    {
        head=stck->top;
        do
        {
            if ( head->exp_data==exp_name )
            {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }

            }
            else
            {
                head=head->next;
            }
        } while (head->next!=NULL);
        /* check agin once more */
        if (head->exp_data==exp_name)
        {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }


        }
    }
    return -999;
}
```

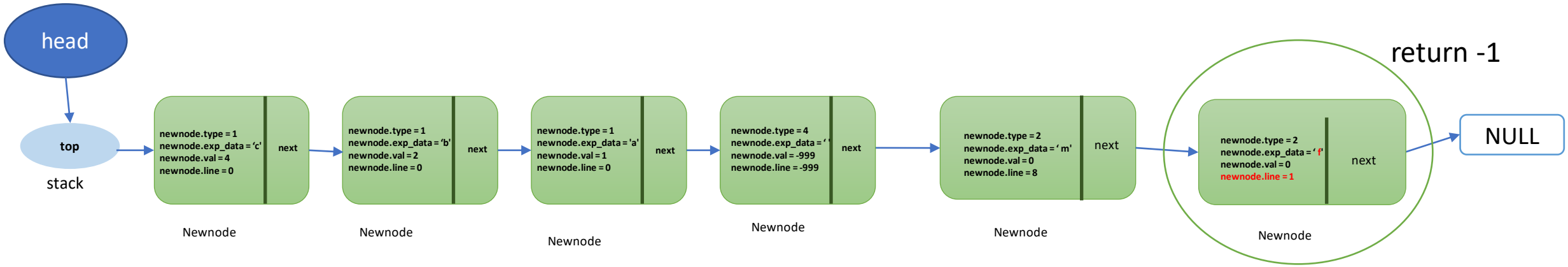
int GetVal(char exp_name,int * line,Stack *stck)

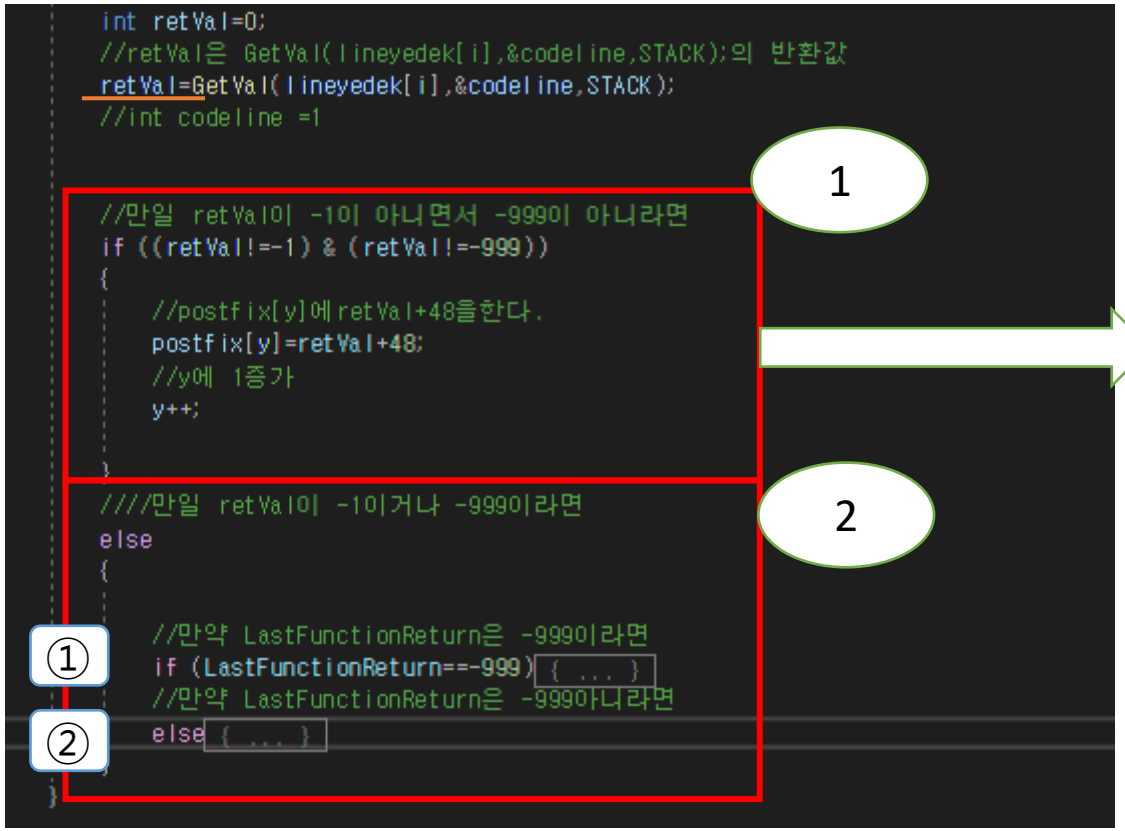
1. 노드 구조체자료형인 head 포인트
2. *line은 0이다.
    1. 만약 stck의 top이 NULL일 경우
        1. ERROR, empty stack... 을 console 출력
    2. 아니면
        1. head는 stck의 top이다.
            1. **do**
                1. 만약 head의 exp_data 가 exp_name와 같다면
                    1. **만약 head의 type이 1이라면**
                        1. **반환값 head의 val**
                    2. 만약 **head의 type이 2**이라면
                        1. *line은 **head의 line**
                        2. **반환값 -1**
                2. 만약 아니라면,
                    1. head는 head의 next
            2. **while(head의 next는 NULL이 아니라면 반복)**

            3. 만약 head의 exp_data 가 exp_name와 같다면
                1. 만약 head의 type이 1이라면
                    1. 반환값 head의 val
                2. 만약 head의 type이 2이라면
                    1. *line은 head의 line
                    2. 반환값 -1
    3. 반환값 -999

예시

lineyedek[i] = f라면





retVal의 값이 2라면

postfix[2] =50

```c
//만일 retVal이 -1이 아니면서 -9999이 아니라면
if ((retVal!=-1) & (retVal!=-999))
{
    /* if variable */
    //postfix[y]에 retVal+48을한다.
    postfix[y]=retVal+48; /* in ascii table numeric values start from 48 */
    //y에 1증가
    y++;

}
//만일 retVal이 -1이거나 -9999이라면
else
{

    //만약 LastFunctionReturn은 -9999이라면
    if (LastFunctionReturn==-999)
    {

    /* if function */
    //정수 j선언
    int j;

    //tempNode.type은 3이다.
    tempNode.type=3;

    //tempNode.line은 curLine이다
    tempNode.line=curLine;
    //STACK은 Push(tempNode,STACK)의 반환값
    STACK=Push(tempNode,STACK);

    /* get function's arguments value */

    //CalingFunctionArgVal은 GetVal(lineyedek[i+2],&dummyint,STACK)의 반환값
    CalingFunctionArgVal=GetVal(lineyedek[i+2],&dummyint,STACK);
```

2

①

```c
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```

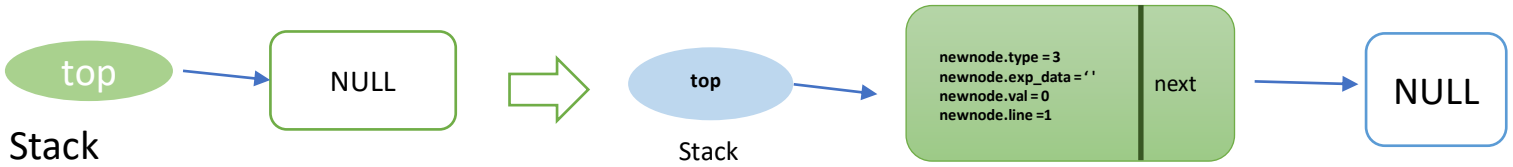
Stack * Push(Node sNode,Stack *stck)

1. Node 자료형의 newnode포인터선언;
2. 만약 newnode는 Node타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   return NULL

3.아니라면

newnode의 type은 sNode의 타입이 된다.
newnode 의 val은 sNode의 val이된다.
newnode 의 exp_data은 sNode의 exp_data이된다.
newnode 의 line 은 sNode의 line이 된다.
newnode 의 next 는 stck의 top이 된다.
stck의 top은 newnode이다

반환값 stck



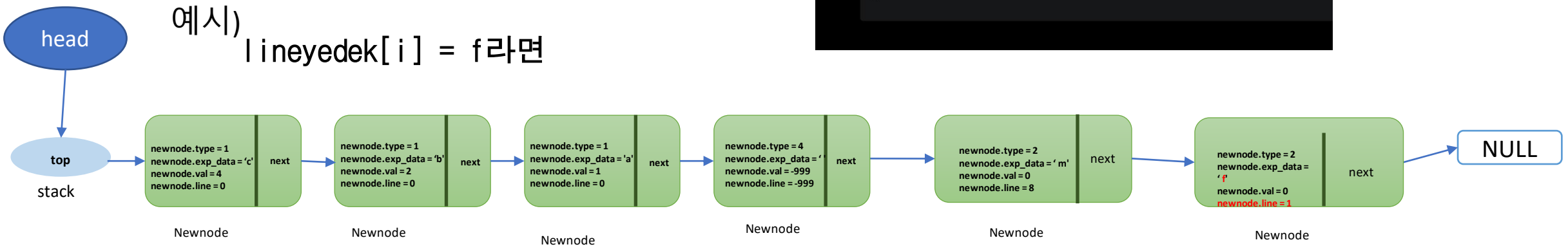
```
//CalingFunctionArgVal은 GetVal(lineyedek[i+2],&dummyint,STACK)의 반환값
CalingFunctionArgVal=GetVal(lineyedek[i+2],&dummyint,STACK);


//filePtr의 파일을 닫는다.
fclose(filePtr);
//filePtr은 argv[1]번째를 읽기 모드로 파일을 연다
filePtr=fopen(argv[1],"r");
//curLine은 0이다.
curLine=0;
/* file reversed to start postion */
/* now go codeline lines to go, to the functions line */


odeline olabilir */
//1부터 codeline보다 작다면 1씩 증가하여 반복
for(j=1;j<codeline;j++)
{
   //filePtr의파일을 최대 4095수까지 읽을 수 있으며, dummy배열에 저장.
   fgets(dummy,4096,filePtr); /* read the file by Line by Line */
   //curLine 1증가
   curLine++;
}

//WillBreak은 1이다.
WillBreak=1;

//while문 나가기
break;
}
```

GetVal

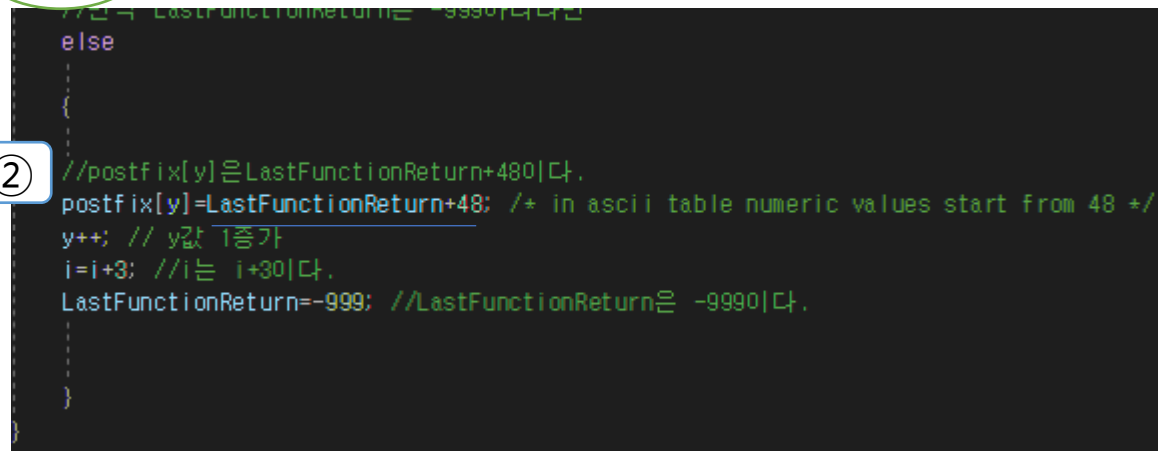
```
int GetVal(char exp_name,int * line,Stack *stck)
{

Node * head;
*line=0;
    if (stck->top == NULL )
    {
        printf("ERROR, empty stack...");
    }
    else
    {
        head=stck->top;
        do
        {
            if ( head->exp_data==exp_name )
            {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }

            }
            else
            {
                head=head->next;
            }
        } while (head->next!=NULL);
        /* check agin once more */
        if (head->exp_data==exp_name)
        {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }


        }
    }

    return -999;
}
```

GetVal()
자료형인 Node인 포인트 head선언

1. head를 stck의 top으로 둔다.
2. head의exp_data== exp_name
3. 와 같다면
   1. head의 type이 1이라면 head의 val이 반환값이다.
   2. head의 type이 2이라면 line은head의 line이다. -1 반환값
4. head의exp_data==exp_name
5. 와 같은게 없다면 head는 head의 next를 해준다. head의 next가 NULL이 아닐때까지 반복해준준다.



2



LastFunctionReturn = 2라면

postfix[3] =50

ⓒ begin과 end가 아니면

③ lineyedek[0]이 '('이라면

WillBreak이 0이라면

```
//WillBreak이 0이라면
if (WillBreak==0)
{

//isStackEmpty(MathStack)이 0이라면 반복
while (isStackEmpty(MathStack)==0)
{

    /* add the poped operator to the postfix */
    //postfix[y]= PopOp(MathStack)의 반환값
    postfix[y]=PopOp(MathStack);
    //y값 1증간
    y++;
}
```

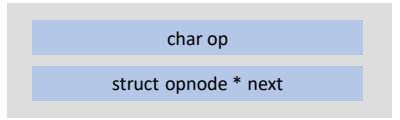
Postfix[2] ='+'

isStackEmpty

```
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)

1. 만약 stck의 top가 0이라면
   1. 반환값 1
2. 아니면 반환값 0



opNode

```
char PopOp(OpStack *opstck)
{
opNode *temp;
char op;
if (opstck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
op=opstck->top->op;
temp=opstck->top;
opstck->top=opstck->top->next;
free(temp);
return op;
}
return NULL;
}
```

PopOp(opstck)

1. opstck의 top이 NULL이면 "Error, empty stack…" 라고 하고 return null;

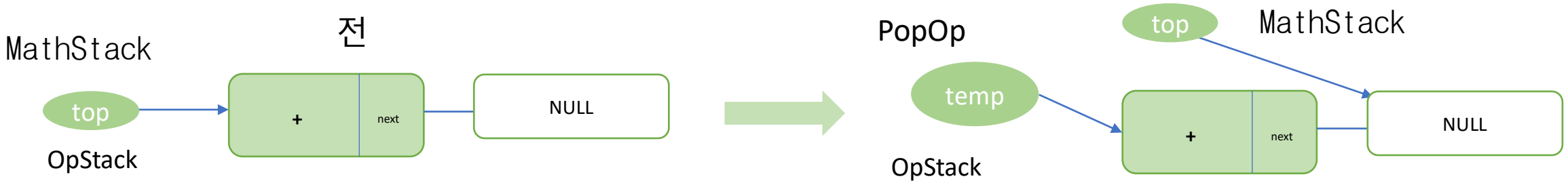
opstck의 top이 NULL이 아니면,

1. op =+ (opstck의 top의 op)를 가리킨다.
2. temp는 opstck의 top이다.
3. opstck의 top은opstck의 top의 next이다.
4. temp를 메모리 해체를한다.
5. return은 +

Op = +

메모리 해제

예)



//postfix[i]이 ₩x0일때까지 반복

```
//postfix[i]이 ₩x0일때까지 반복
while(postfix[i]!='₩x0')
{
    //postfix[i]가 숫자라면
    if (isdigit(postfix[i])) {
        /* push to stack */
        //CalcStack PushPostfix(postfix[i]-'0',CalcStack);의 반환값
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    //postfix[i]가 +이거나 postfix[i]가 -이거나 postfix[i]가 *이거나 postfix[i]가 / 이라면
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
        //val1 = PopPostfix(CalcStack)의 반환값
        val1=PopPostfix(CalcStack);

        //val2 = PopPostfix(CalcStack)의 반환값
        val2=PopPostfix(CalcStack);

        //postfix[i] 가
        switch (postfix[i])
        {
            //+일때 resultVal=val2+val1
            //후 break문으로 switch문 빠져나간다.
            case '+': resultVal=val2+val1;break;
            //- 일때 resultVal=val2-val1
            //후 break문으로 switch문 빠져나간다
            case '-': resultVal=val2-val1;break;

            ///일때 resultVal=val2/val1
            //후 break문으로 switch문 빠져나간다
            case '/': resultVal=val2/val1;break;
            //*일때 resultVal=val2*val1
            // //후 break문으로 switch문 빠져나간다
            case '*': resultVal=val2*val1;break;
        }

        //CalcStack = PushPostfix(resultVal,CalcStack)의 반환값
        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    // i에 1 증가
    i++;
}
```

1

2

3

```
PostfixStack * PushPostfix(int val,PostfixStack *poststck)
{
Postfixnode *newnode;
if ((newnode=(Postfixnode*)malloc(sizeof(Postfixnode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->val=val;
newnode->next=poststck->top;
poststck->top=newnode;
return poststck;
}
}
```

PostfixStack * PushPostfix(int val,PostfixStack *poststck)

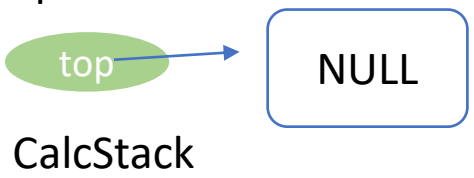
1. Postfixnode구조체의 자료형인 newnode 포인터 선언
2. 만약 newnode는 Postfixnode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   NULL반환
3. 아니면라면,
   1. newnode의 val 은 매개변수 val이다
   2. newnode의 next 는 poststck의 top이다.
   3. poststck의 top 는 newnode이다.
   4. 반환값 poststck

```
//postfix[i]이 ₩x0일때까지 반복
while(postfix[i]!='₩x0')
{
    //postfix[i]가 숫자라면
    if (isdigit(postfix[i])) {
        /* push to stack */
        //CalcStack PushPostfix(postfix[i]-'0',CalcStack);의 반환값
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    //postfix[i]가 +이거나 postfix[i]가 -이거나 postfix[i]가 *이거나 postfix[i]가 / 이라면
```

1

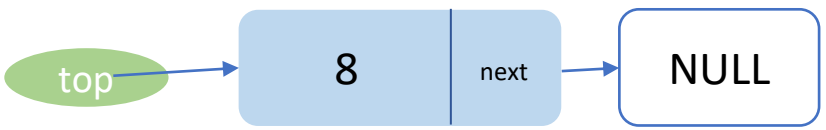
```
PostfixStack * PushPostfix(int val,PostfixStack *poststck)
{
Postfixnode *newnode;
if ((newnode=(Postfixnode*)malloc(sizeof(Postfixnode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->val=val;
newnode->next=poststck->top;
poststck->top=newnode;
return poststck;
}
}
```

pushPostfix 전



CalcStack

pushPostfix 후



CalcStack

newnode

PostfixStack * PushPostfix(int val,PostfixStack *poststck)

1. Postfixnode구조체의 자료형인 newnode 포인터 선언
2. 만약 newnode는 Postfixnode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   NULL반환
3. 아니면라면,
   1. newnode의 val 은 매개변수 val이다
   2. newnode의 next 는 poststck의 top이다.
   3. poststck의 top 는 newnode이다.
   4. 반환값 poststck

```
i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
①       val1=PopPostfix(CalcStack);

②       val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

예) val1 = 2, val2 =6이라면

val1 = 2
val2=6

resultVal = 6+2

2

Postfix[]

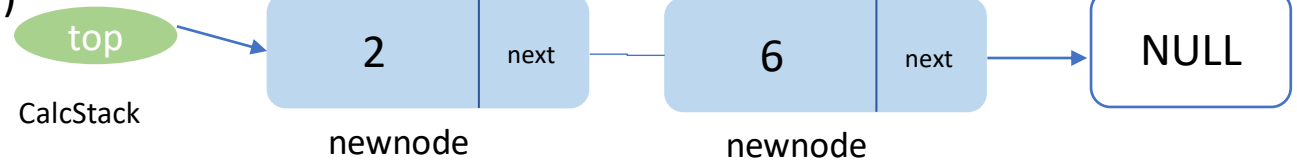
54	50	'+'	52	'/'	\0

```
char PopPostfix(PostfixStack *poststck)
{
Postfixnode *temp;
int val;
if (poststck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
val=poststck->top->val;
temp=poststck->top;
poststck->top=poststck->top->next;
free(temp);
return val;
}
return NULL;
}
```

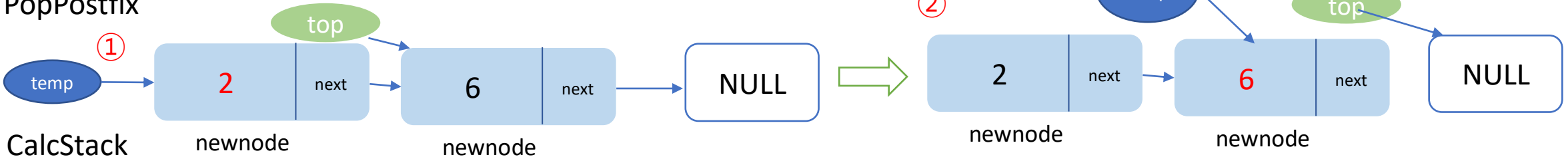
char PopPostfix(PstfixStack *poststcPostfixnode구조체의 자료k)

1. 형 temp 포인터 선언
2. 정수자료형인 val선언
3. 만약 poststck의 top이 NULL이라면
   1. ERROR, empty stack...console에 출력
4. 아니라면,
   1. val 은 poststck의 top의 val이다.
   2. tmep는 poststck의 top
   3. poststck의 top 은 poststck의 top의 next이다.
   4. temp의 동적메모리 해제
   5. 반환값 val
5. 반환값 NULL

예)



PopPostfix



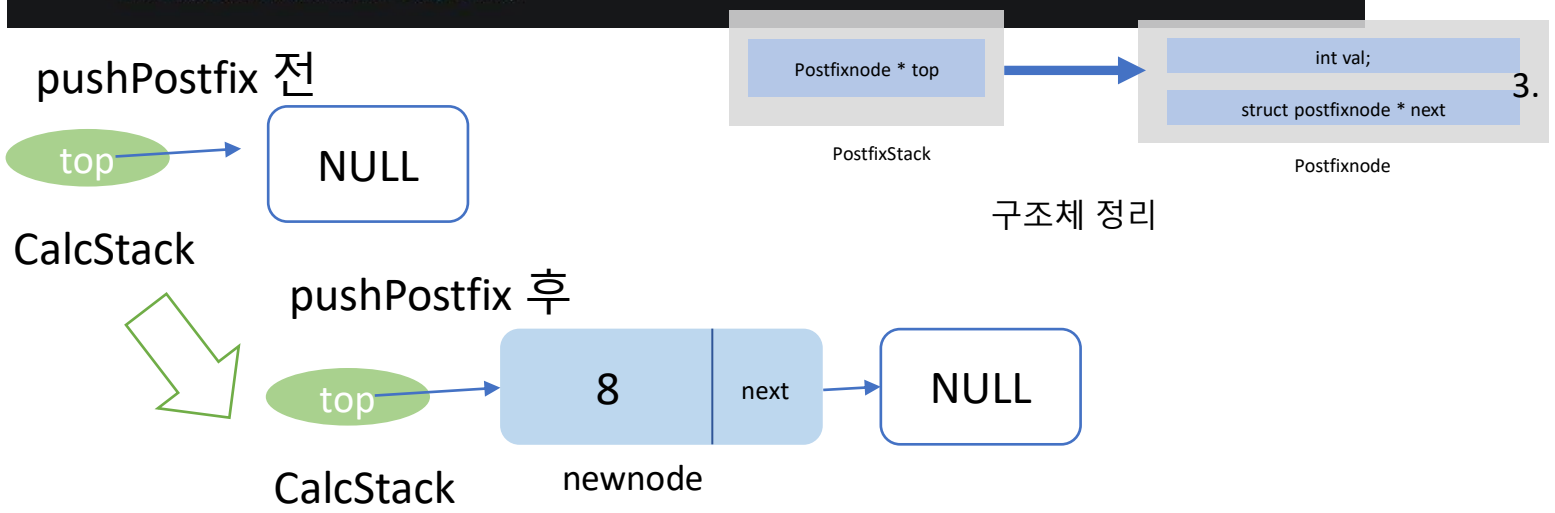
```
i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
        val1=PopPostfix(CalcStack);

        val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

pushPostfix(8,CalcStck)

```
PostfixStack * PushPostfix(int val,PostfixStack *poststck)
{
Postfixnode *newnode;
if ((newnode=(Postfixnode*)malloc(sizeof(Postfixnode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->val=val;
newnode->next=poststck->top;
poststck->top=newnode;
return poststck;
}
}
```

PostfixStack * PushPostfix(int val,PostfixStack *poststck)

1. Postfixnode구조체의 자료형인 newnode 포인터 선언
2. 만약 newnode는 Postfixnode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   NULL반환
3. 아니면라면,
   1. newnode의 val 은 매개변수 val이다
   2. newnode의 next 는 poststck의 top이다.
   3. poststck의 top 는 newnode이다.
   4. 반환값 poststck



구조체 정리

```
//filePtr 파일 닫기
fclose(filePtr);
//printAllStack(STACK);

//STACK은FreeAll(STACK)의 반환값
STACK=FreeAll(STACK);

//Press a key to exit...콘솔에 출력
printf("₩nPress a key to exit...");
//키를 입력받는다.(input)
getch();

//반환값 0
return  0;
```

1.filePtr 파일 닫기
2.STACK= NULL
3. " Press a key to exit..."라고 콘솔에 나온다.
4. 키를 입력받는다.(input)
5. return 0

FreeAll

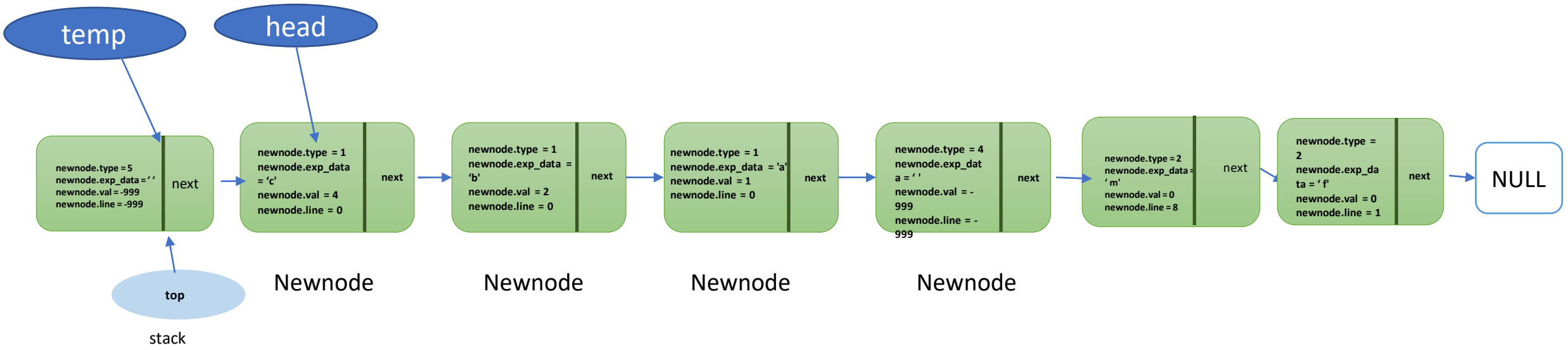
```
Stack * FreeAll(Stack * stck)
{
Node * temp;
Node * head;

  if (stck->top != NULL )
  {
    head=stck->top;
    do
    {

      temp=head;
      head=head->next;
      free(temp);

    } while (head->next!=NULL);
  }

return NULL;
}
```

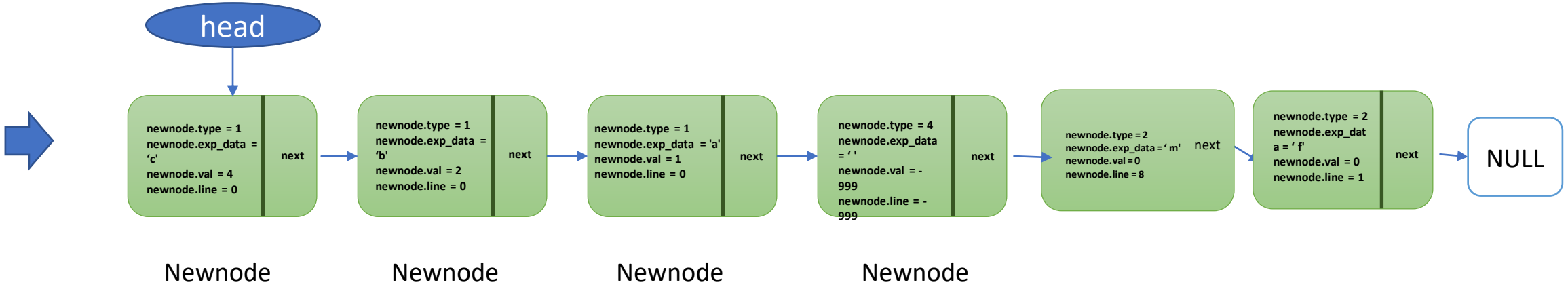
Stack * FreeAll(Stack * stck)

1. Node구조체 자료형인 포인터 temp
2. Node구조체 자료형인 포인터 head
3. 만약 stck의 top이 NULL이 아니라면,
    1. head는 stck의top이다.
    2. do
        1. temp는 head이다.
        2. head는 head의 next이다.
        3. temp 동적 메모리 할당 해제한다.
    3. while (head의 next가 NULL이 아니라면 반복)
4. 반환값 NULL

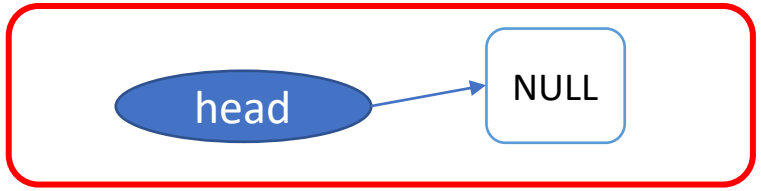
예시)

FreeAll()





head의 next가 NULL이될때까지 반복 ...



실전

inpu1.sql

파일 읽기

```
//파일 스트림의 끝을 만날 때까지 반복
while (!feof(filePtr))
{

    int k=0;

    //입력 스트림에서 문자열 읽기

    fgets(line,4096, filePtr); /* read the file by Line by Line */
    /* scan for /t characters, get rid of them! */
    while(line[k]!='₩0')
    {
     if (line[k]=='₩t')
     {
        line[k]=' ';
     }

     k++;
    }

    strcpy(lineyedek,line);

    curLine++;
    tempNode.val=-999;
    tempNode.exp_data=' ';
    tempNode.line=-999;
    tempNode.type=-999;



    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }
    else{ ... }


}
```

1. 파일 스트림의 끝을 만날 때까지 반복
2. 정수 k의 값은 0
3. fget()
   fgets(파일 데이터를 저장할 변수, 읽어들일 최대 문자 수, 읽을 파일)

filePtr의파일을 최대 4095수까지 읽고 line배열에 저장.

4. whil문 line[k]이 null 이 아닐 시때까지 반복.
   4-1) 만약 line[k]가 \t(탭키)일 시, line[k]는 ‘ ‘으로 바꾼다.
   k증가

5. line을 lineyedek에 문자열 복사
6. curLine을 1증가
   tempNode.val= -999; // tempNode.val값을 -999로 대입
   tempNode.exp_data=' '; // tempNode.exp_data값을 ‘ ‘로 대입
   tempNode.line=-999;// tempNode.line 을 -999로 대입
   tempNode.type=-999;// tempNode.type 을 -999로 대입

7. 3가지로 나누어짐
   ① line이 begin일 경우
   ② line이 end일 경우
   ③ line이 begin과 end가 아닐경우

```
//파일 스트림의 끝을 만날 때까지 반복
while (!feof(filePtr))
{

    int k=0;

    //입력 스트림에서 문자열 읽기

    fgets(line,4096, filePtr); /* read the file by Line by Line */
    /* scan for /t characters. get rid of them! */
    while(line[k]!='₩0')
    {
     if (line[k]=='₩t')
     {
        line[k]=' ';
     }

     k++;
    }

    strcpy(lineyedek,line);

    curLine++;
    tempNode.val=-999;
    tempNode.exp_data=' ';
    tempNode.line=-999;
    tempNode.type=-999;



    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }
    else{ ... }


}
```

input1.sql(읽을 파일)

```
function f(int a)
begin
    int b = 6;
    int c = 2;
    ((b+c)/a);
end

function main()
begin
    int a = 1;
    int b = 2;
    int c = 4;
    ((6 + f(c) ) / b);
end
```

1. 파일 스트림의 끝을 만날 때까지 반복
2. 정수 k의 값은 0
3. fget을 통해 function f(int a)을 읽어 line에 저장 .
4. whil문 line[k]이 null 이 아닐 시때까지 반복.
   4-1) 만약 line[k]가 \t(탭키)일 시, line[k]는 ' '으로 바꾼다.
   k에 1증가
5. line을 lineyedek에 문자열 복사
6. curLine을 1증가

tempNode.val= -999; // tempNode.val값을 -999로 대입
tempNode.exp_data=' '; // tempNode.exp_data값을 ' '로 대입
tempNode.line=-999;// tempNode.line 을 -999로 대입
tempNode.type=-999;// tempNode.type 을 -999로 대입

```
//파일 스트림의 끝을 만날 때까지 반복
while (!feof(filePtr))
{

    int k=0;

    //입력 스트림에서 문자열 읽기

    fgets(line,4096, filePtr); /* read the file by Line by Line */
    /* scan for /t characters. get rid of them! */
    while(line[k]!='₩0')
    {
     if (line[k]=='₩t')
     {
        line[k]=' ';
     }

     k++;
    }

    strcpy(lineyedek,line);

    curLine++;
    tempNode.val=-999;
    tempNode.exp_data=' ';
    tempNode.line=-999;
    tempNode.type=-999;



    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }
    else{ ... }


}
```

curLine =2

input1.sql(읽을 파일)

```
function f(int a)
begin
   int b = 6;
   int c = 2;
   ((b+c)/a);
end

function main()
begin
   int a = 1;
   int b = 2;
   int c = 4;
   ((6 + f(c) ) / b);
end
```

1. 파일 스트림의 끝을 만날 때까지 반복
2. 정수 k의 값은 0
3. fget을 통해 begin 을 읽어 line에 저장 .
4. whil문 line[k]이 null 이 아닐 시때까지 반복.
   4-1) 만약 line[k]가 \t(탭키)일 시, line[k]는 ‘ ‘으로 바꾼다.
   k증가
5. line을 lineyedek에 문자열 복사
6. curLine을 1증가

tempNode.val= -999; // tempNode.val값을 -999로 대입
tempNode.exp_data=' ‘; // tempNode.exp_data값을 ‘ ‘로 대입
tempNode.line=-999;// tempNode.line 을 -999로 대입
tempNode.type=-999;// tempNode.type 을 -999로 대입

① line이 begin일 경우

```
// @하나
        //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
        //begind이면
        if (!strcmpi("begin\n",line) | !strcmpi("begin",line))
        {
            //foundMain == 0이면 false 0이 아니면 True
            if (foundMain)
            {
                tempNode.type=4;
                STACK=Push(tempNode,STACK);
            }
        }
```

int foundMain=0임으로 false

```
//파일 스트림의 끝을 만날 때까지 반복
while (!feof(filePtr))
{

    int k=0;

    //입력 스트림에서 문자열 읽기

    fgets(line,4096, filePtr); /* read the file by Line by Line */
    /* scan for /t characters. get rid of them! */
    while(line[k]!='₩0')
    {
    if (line[k]=='₩t')
    {
       line[k]=' ';
    }

    k++;
    }

    strcpy(lineyedek,line);

    curLine++;
    tempNode.val=-999;
    tempNode.exp_data=' ';
    tempNode.line=-999;
    tempNode.type=-999;



    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }
    else{ ... }


}
```

curLine=3

input1.sql(읽을 파일)

```
function f(int a)
begin
   int b = 6;
   int c = 2;
   ((b+c)/a);
end

function main()
begin
   int a = 1;
   int b = 2;
   int c = 4;
   ((6 + f(c) ) / b);
end
```

1. 파일 스트림의 끝을 만날 때까지 반복
2. 정수 k의 값은 0
3. fget을 통해 int b = 6; 을 읽어 line에 저장 .
4. whil문 line[k]이 null 이 아닐 시때까지 반복.
   4-1) 만약 line[k]가 \t(탭키)일 시, line[k]는 ' '으로 바꾼다.
   k증가
5. line을 lineyedek에 문자열 복사
6. curLine을 1증가

tempNode.val= -999; // tempNode.val값을 -999로 대입
tempNode.exp_data=' '; // tempNode.exp_data값을 ' '로 대입
tempNode.line=-999;// tempNode.line 을 -999로 대입
tempNode.type=-999;// tempNode.type 을 -999로 대입

③ line이 begin과 end가 아닐경우

```
else
{
    //we need to tokinize
    //int
    // 공백 -> \0으로 바꿈
    firstword=strtok(line," ");

    //1.
    //int가 firstword이라면
    if (!strcmpi("int",firstword))
    {
        //(1)-1
        //foundMain ==0
        //foundMain == 0이면 false 0이 아니면 True
        if (foundMain){ ... }
    }
    //2.
```

firstword =int

int foundMain=0임으로 false

```
//파일 스트림의 끝을 만날 때까지 반복
while (!feof(filePtr))
{

    int k=0;

    //입력 스트림에서 문자열 읽기

    fgets(line,4096, filePtr); /* read the file by Line by Line */
    /* scan for /t characters. get rid of them! */
    while(line[k]!='₩0')
    {
    if (line[k]=='₩t')
    {
       line[k]=' ';
    }

    k++;
    }

    strcpy(lineyedek,line);

    curLine++;
    tempNode.val=-999;
    tempNode.exp_data=' ';
    tempNode.line=-999;
    tempNode.type=-999;



    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }
    else{ ... }


}
```

curLine=4

input1.sql(읽을 파일)

```
function f(int a)
begin
    int b = 6;
    int c = 2;
   ((b+c)/a);
end

function main()
begin
    int a = 1;
    int b = 2;
    int c = 4;
   ((6 + f(c) ) / b);
end
```

1. 파일 스트림의 끝을 만날 때까지 반복
2. 정수 k의 값은 0
3. fget을 통해 int c = 2;을 읽어 line에 저장 .
4. whil문 line[k]이 null 이 아닐 시때까지 반복.
   4-1) 만약 line[k]가 \t(탭키)일 시, line[k]는 ' '으로 바꾼다.
   k증가
5. line을 lineyedek에 문자열 복사
6. curLine을 1증가

tempNode.val= -999; // tempNode.val값을 -999로 대입
tempNode.exp_data=' '; // tempNode.exp_data값을 ' '로 대입
tempNode.line=-999;// tempNode.line 을 -999로 대입
tempNode.type=-999;// tempNode.type 을 -999로 대입

③ line이 begin과 end가 아닐경우

```
else
{
    //we need to tokinize
    //int
    // 공백 -> \0으로 바꿈
    firstword=strtok(line," ");

    //1.
    //int가 firstword이라면
    if (!strcmpi("int",firstword))
    {
        //(1)-1
        //foundMain ==0
        //foundMain == 0이면 false 0이 아니면 True
        if (foundMain){ ... }
    }
    //2.
```

firstword =int

int foundMain=0임으로 false

```
//파일 스트림의 끝을 만날 때까지 반복
while (!feof(filePtr))
{

    int k=0;

    //입력 스트림에서 문자열 읽기

    fgets(line,4096, filePtr); /* read the file by Line by Line */
    /* scan for /t characters. get rid of them! */
    while(line[k]!='₩0')
    {
    if (line[k]=='₩t')
    {
       line[k]=' ';
    }

    k++;
    }

    strcpy(lineyedek,line);

    curLine++;
    tempNode.val=-999;
    tempNode.exp_data=' ';
    tempNode.line=-999;
    tempNode.type=-999;



    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }
    else{ ... }


}
```

curLine=5

input1.sql(읽을 파일)

```
function f(int a)
begin
    int b = 6;
    int c = 2;
    ((b+c)/a);
end

function main()
begin
    int a = 1;
    int b = 2;
    int c = 4;
    ((6 + f(c) ) / b);
end
```

1. 파일 스트림의 끝을 만날 때까지 반복
2. 정수 k의 값은 0
3. fget을 통해 int ((b+c)/a);을 읽어 line에 저장 .
4. whil문 line[k]이 null 이 아닐 시때까지 반복.
   4-1) 만약 line[k]가 \t(탭키)일 시, line[k]는 ' '으로 바꾼다.
   k증가
5. line을 lineyedek에 문자열 복사
6. curLine을 1증가

tempNode.val= -999; // tempNode.val값을 -999로 대입
tempNode.exp_data=' '; // tempNode.exp_data값을 ' '로 대입
tempNode.line=-999;// tempNode.line 을 -999로 대입
tempNode.type=-999;// tempNode.type 을 -999로 대입

③ line이 begin과 end가 아닐경우

```
//ⓒ begin과 end가 아니면
else
{
    //we need to tokinize
    //int
    // 공백 -> \0으로 바꿈
    firstword=strtok(line," ");

    //1.
    //int가 firstword이라면
    if (!strcmpi("int",firstword))
    {
        //(1)-1
        //foundMain ==0
        //foundMain == 0이면 false 0이 아니면 True
        if (foundMain) { ... }
    }
    //2.
    else if (!strcmpi("function",firstword)) { ... }

    //3
    else if (firstword[0]=='(')
    {
    //(3)-1
    if (foundMain) { ... }
    }
}
```

firstword =(

int foundMain=0임으로 false

end

```
//파일의 끝에 도달했는지 여부를 확인
//파일의 끝에 도달하지 못했을 경우에 0 반환
//파일 스트림의 끝을 만날 때까지 반복
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //fget을 통해 function f(int a)을 읽어 line에 저장 ,
    fgets(line,4096,filePtr); /* read the file by Line by Line */
    /* scan for /t characters, get rid of them! */
    // while문 line[k]이  null 이 아닐 시때까지 반복.

    while(line[k]!='₩0')
    {
      //  4-1) 만약 line[k]가 ₩t(탭키)일 시,
     if (line[k]=='₩t')
     {
       //line[k]는  ' '으로 바꾼다.
      line[k]=' ';
     }
     //k에 1증가
     k++;
    }
    //5, line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을  ' '로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //begin과 line이 동일하다면
    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }


    //ⓑ
    //  //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //line이 end라면
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }

    //ⓒ begin과 end가 아니면
    else{ ... }


}
```

curLine=6

input1.sql(읽을 파일)

```
function f(int a)
begin
    int b = 6;
    int c = 2;
    ((b+c)/a);
end

function main()
begin
    int a = 1;
    int b = 2;
    int c = 4;
    ((6 + f(c) ) / b);
end
```

```
//ⓑ
//  //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
//line이 end라면
else if (!strcmpi("end₩n",line) | !strcmpi("end",line) )
{
    //(1)

    //foundMain == 0이면  false 0이 아니면 True
    if (foundMain){ ... }
}
```

function main()

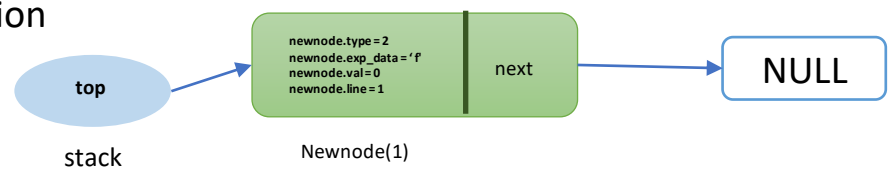
```
//파일의 끝에 도달했는지 여부를 확인
//파일의 끝에 도달하지 못했을 경우에 0 반환
//파일 스트림의 끝을 만날 때까지 반복
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //fget을 통해 function f(int a)을 읽어 line에 저장 .
    fgets(line,4096,filePtr); /* read the file by Line by Line */
    /* scan for /t characters, get rid of them! */
    // while문 line[k]이  null 이 아닐 시때까지 반복.

    while(line[k]!='₩0')
    {
       //  4-1) 만약 line[k]가 ₩t(탭키)일 시,
     if (line[k]=='₩t')
     {
        //line[k]는  ' '으로 바꾼다.
       line[k]=' ';
     }
      //k에 1증가
     k++;
    }
    //5. line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을  ' '로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //begin과 line이 동일하다면
    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)) { ... }


    //ⓑ
    //  //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //line이 end라면
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ) { ... }

    //ⓒ begin과 end가 아니면
    else { ... }


}
```

curLine=8

input1.sql(읽을 파일)

```
function f(int a)
begin
    int b = 6;
    int c = 2;
    ((b+c)/a);
end

function main()
begin
    int a = 1;
    int b = 2;
    int c = 4;
    ((6 + f(c) ) / b);
end
```

function



```
else if (!strcmpi("function",firstword))
{
  //tempNode.type는 2이다
  tempNode.type=2;

  //  //자른 문자 다음부터 구분자 찾기
  firstword=strtok(NULL," ");

  //tempNode.exp_data는 firstword[0]이다
  tempNode.exp_data=firstword[0];

  //tempNode.line 는 curLine이다
  tempNode.line=curLine;
  //tempNode.val은 0이다.
  tempNode.val=0;
  //STACK은 Push(tempNode,STACK); 에서의 반환값이다.
  STACK=Push(tempNode,STACK);

  //main
  //(2)-1
  //만약 firstword배열의 0번째가 'm'이며 firstword배열의 1번째가 'a'이며 firstword배열의 2번째가 'i'이며 firstword배열의 3번째가 'n'이라면
  if ( (firstword[0]=='m') & (firstword[1]=='a') & (firstword[2]=='i') & (firstword[3]=='n') )
  {
    /*printf("Found function main() in line %d. Starting to running the script...\n",curLine);*/
    //foundMain 은 1이다.
    foundMain=1;

  }
```

tempNode.type=2

firstword = main

tempNode.exp_data = m

tempNode.line=8

tempNode. val=0

foundMain =1

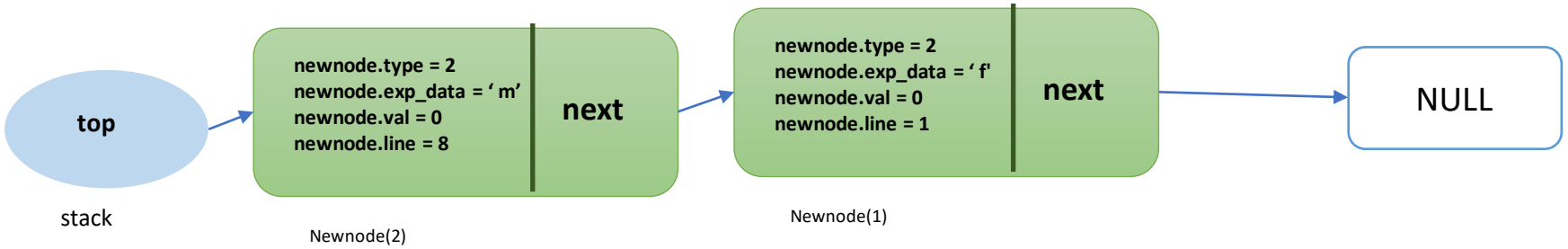
Push

```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```

Node자료형의newnode포인터선언;
만약 newnode는 Node타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우 ERROR, Couldn't allocate memory... 출력
NULL리턴

newnode.type = 2
newnode.val = 0
newnode.exp_data = ' m'
newnode.line = 8
newnode 의 next 는 stck의 top이 된다.
stck의 top은 newnode이다
반환값 stck

function



```
//파일의 끝에 도달했는지 여부를 확인
//파일의 끝에 도달하지 못했을 경우에 0 반환
//파일 스트림의 끝을 만날 때까지 반복
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //fget을 통해 function f(int a)을 읽어 line에 저장 .
    fgets(line,4096,filePtr); /* read the file by Line by Line */
    /* scan for /t characters, get rid of them! */
    // while문 line[k]이  null 이 아닐 시때까지 반복.

    while(line[k]!='₩0')
    {
       //  4-1) 만약 line[k]가 ₩t(탭키)일 시,
     if (line[k]=='₩t')
     {
        //line[k]는  ' '으로 바꾼다.
       line[k]=' ';
     }
      //k에 1증가
     k++;
    }
    //5, line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을  ' '로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //begin과 line이 동일하다면
    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }


    //ⓑ
    //  //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //line이 end라면
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }

    //ⓒ begin과 end가 아니면
    else{ ... }


}
```

curLine=9

input1.sql(읽을 파일)

```
function f(int a)
begin
    int b = 6;
    int c = 2;
    ((b+c)/a);
end

function main()
begin
    int a = 1;
    int b = 2;
    int c = 4;
    ((6 + f(c) ) / b);
end
```

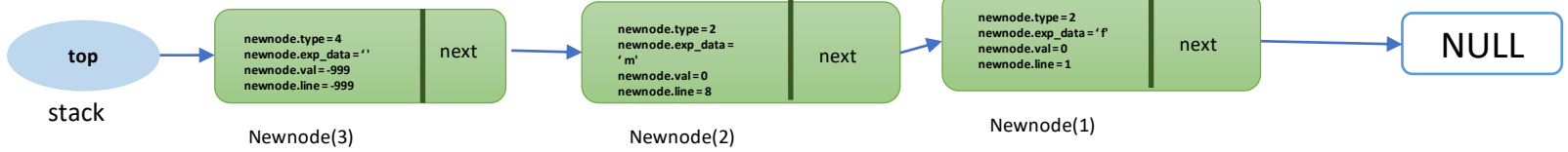
```
// ⓐ하나
//strcmpi :  둘이 동일하면 1(true) 반환하고 동일하지 않으면 0(false)
//begin과 line이 동일하다면
if (!strcmpi("beginMn",line) | !strcmpi("begin",line))
{
    //foundMain == 0이면 false 0이 아니면 True
    if (foundMain)
    {
        //tempNode.type 는 4이다.
        tempNode.type=4;
        ////STACK은 Push(tempNode,STACK); 에서의 반환값이다.
        STACK=Push(tempNode,STACK);
    }
}
```

Push

```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```

Node자료형의newnode포인터선언;
만약 newnode는 Node타입크기 만큼메모리를 할당받으며,
newnode가NULL일 경우 ERROR, Couldn't allocate memory... 출력
NULL리턴

newnode.type = 4
newnode.val = -999
newnode.exp_data = ' '
newnode.line = -999
newnode 의 next 는 stck의 top이 된다.
stck의 top은 newnode이다
반환값 stck



int a = 1;

```
//파일의 끝에 도달했는지 여부를 확인
//파일의 끝에 도달하지 못했을 경우에 0 반환
//파일 스트림의 끝을 만날 때까지 반복
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //fget을 통해 function f(int a)을 읽어 line에 저장 .
    fgets(line,4096,filePtr); /* read the file by Line by Line */
    /* scan for /t characters, get rid of them! */
    // whil문 line[k]이  null 이 아닐 시때까지 반복.

    while(line[k]!='₩0')
    {
       //  4-1) 만약 line[k]가 ₩t(탭키)일 시,
     if (line[k]='₩t')
     {
        //line[k]는  ' '으로 바꾼다.
       line[k]=' ';
     }
      //k에 1증가
     k++;
    }
    //5, line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을  ' '로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //begin과 line이 동일하다면
    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }


    //ⓑ
    //  //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //line이 end라면
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }

    //ⓒ begin과 end가 아니면
    else{ ... }


}
```

curLine=10

input1.sql(읽을 파일)

```
function f(int a)
begin
    int b = 6;
    int c = 2;
    ((b+c)/a);
end

function main()
begin
    int a = 1;
    int b = 2;
    int c = 4;
    ((6 + f(c) ) / b);
end
```

int

```
// 공백 -> 0으로 바꿈
firstword=strtok(line," ");

//1.
//int가 firstword이라면
if (!strcmpi("int",firstword))
{
    //(1)-1
    //foundMain ==0
    //foundMain == 0이면 false 0이 아니면 True
    if (foundMain)
    {
        //ttempNode.type 은 1
        tempNode.type=1; /*integer*/

        //자른 문자 다음부터 구분자 또 찾기
        firstword=strtok(NULL," ");

        //tempNode.exp_data의 exp_data = firstword[0]
        tempNode.exp_data=firstword[0];


        //자른 문자 다음부터 구분자  찾기
        firstword=strtok(NULL," ");

        //(2)-1
        /* check for = */
        //firstword가 '='이라면 true
        if (!strcmpi("=",firstword))
        {
            //firstword는 자른 문자 다음부터 구분자  찾기은 값이다.
            firstword=strtok(NULL," ");
        }

        //문자열을 정수 타입 변화 ex) 1
        //tempNode의 val = firstword를 정수 변환
        tempNode.val=atoi(firstword);
        //tempNode의 line은  0
        tempNode.line=0;
        //STACK은 Push(tempNode,STACK)한 값
        STACK=Push(tempNode,STACK);
    }
}
```

firstword=int

firstword=a

firstword= “=“

tempNode.val = 2

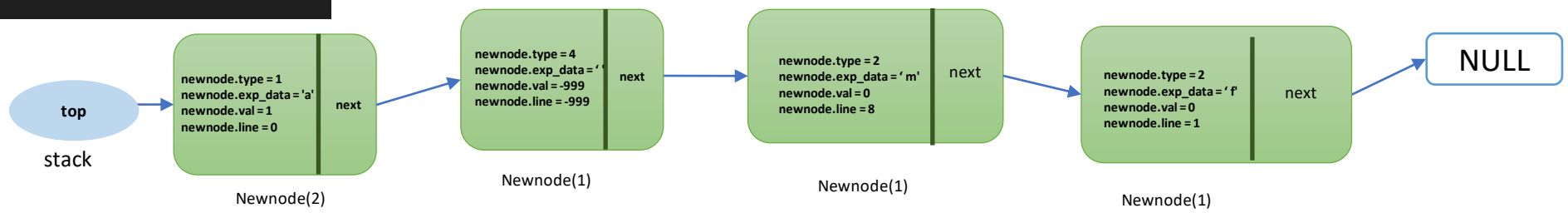
tempNode.line = 0

Push

```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```

Node자료형의newnode포인터선언;
만약 newnode는 Node타입크기 만큼메모리를 할당받으며,
newnode가NULL일 경우 ERROR, Couldn't allocate memory... 출력
NULL리턴

newnode.type = 1
newnode.val = 1
newnode.exp_data = ‘b’
newnode.line = 0
newnode 의 next 는 stck의 top이 된다.
stck의 top은 newnode이다
반환값 stck



```
//파일의 끝에 도달했는지 여부를 확인
//파일의 끝에 도달하지 못했을 경우에 0 반환
//파일 스트림의 끝을 만날 때까지 반복
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //fget을 통해 function f(int a)을 읽어 line에 저장 .
    fgets(line,4096,filePtr); /* read the file by Line by Line */
    /* scan for /t characters, get rid of them! */
    // while문 line[k]이  null 이 아닐 시때까지 반복.

    while(line[k]!='₩0')
    {
       //  4-1) 만약 line[k]가 ₩t(탭키)일 시,
     if (line[k]=='₩t')
     {
        //line[k]는  ' '으로 바꾼다.
       line[k]=' ';
     }
      //k에 1증가
     k++;
    }
    //5, line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을  ' '로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //begin과 line이 동일하다면
    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }


    //ⓑ
    //  //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //line이 end라면
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }

    //ⓒ begin과 end가 아니면
    else{ ... }


}
```

curLine=11

input1.sql(읽을 파일)

```
function f(int a)
begin
    int b = 6;
    int c = 2;
    ((b+c)/a);
end

function main()
begin
    int a = 1;
    int b = 2;
    int c = 4;
    ((6 + f(c) ) / b);
end
```

int

```
// 공백 -> 0으로 바꿈
firstword=strtok(line," ");

//1.
//int가 firstword이라면
if (!strcmpi("int",firstword))
{
    //(1)-1
    //foundMain ==0
    //foundMain == 0이면 false 0이 아니면 True
    if (foundMain)
    {
        //ttempNode.type 은 1
        tempNode.type=1; /*integer*/

        //자른 문자 다음부터 구분자 또 찾기
        firstword=strtok(NULL," ");

        //tempNode.exp_data의 exp_data = firstword[0]
        tempNode.exp_data=firstword[0];


        //자른 문자 다음부터 구분자  찾기
        firstword=strtok(NULL," ");

        //(2)-1
        /* check for = */
        //firstword가 '='이라면 true
        if (!strcmpi("=",firstword))
        {
            //firstword는 자른 문자 다음부터 구분자  찾기은 값이다
            firstword=strtok(NULL," ");
        }

        //문자열을 정수 타입 변화 ex) 1
        //tempNode의 val = firstword를 정수 변환
        tempNode.val=atoi(firstword);
        //tempNode의 line은  0
        tempNode.line=0;
        //STACK은 Push(tempNode,STACK)한 값
        STACK=Push(tempNode,STACK);
    }
}
```

firstword=int

firstword=b

firstword= “=“

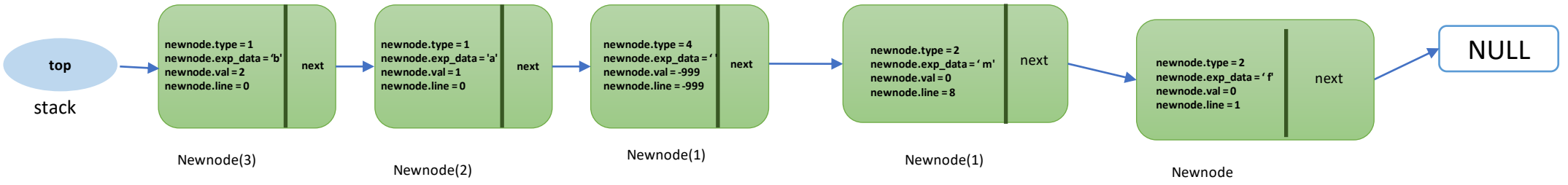
tempNode.val = 1

Push

```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```

Node자료형의newnode포인터선언;
만약 newnode는 Node타입크기 만큼메모리를 할당받으며,
newnode가NULL일 경우 ERROR, Couldn't allocate memory... 출력
NULL리턴

newnode.type = 1
newnode.val = 1
newnode.exp_data = ‘b’
newnode.line = 0
newnode 의 next 는 stck의 top이 된다.
stck의 top은 newnode이다
반환값 stck



```
//파일의 끝에 도달했는지 여부를 확인
//파일의 끝에 도달하지 못했을 경우에 0 반환
//파일 스트림의 끝을 만날 때까지 반복
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //fget을 통해 function f(int a)을 읽어 line에 저장 .
    fgets(line,4096,filePtr); /* read the file by Line by Line */
    /* scan for /t characters, get rid of them! */
    // while문 line[k]이  null 이 아닐 시때까지 반복.

    while(line[k]!='₩0')
    {
      //  4-1) 만약 line[k]가 ₩t(탭키)일 시,
     if (line[k]=='₩t')
     {
       //line[k]는  ' '으로 바꾼다.
      line[k]=' ';
     }
     //k에 1증가
     k++;
    }
    //5, line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을  ' '로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //begin과 line이 동일하다면
    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }


    //ⓑ
    //  //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //line이 end라면
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }

    //ⓒ begin과 end가 아니면
    else{ ... }


}
```

curLine=12

input1.sql(읽을 파일)

```
function f(int a)
begin
    int b = 6;
    int c = 2;
    ((b+c)/a);
end

function main()
begin
    int a = 1;
    int b = 2;
    int c = 4;
    ((6 + f(c) ) / b);
end
```

int

```
// 공백 -> 0으로 바꿈
firstword=strtok(line," ");

//1.
//int가 firstword이라면
if (!strcmpi("int",firstword))
{
    //(1)-1
    //foundMain ==0
    //foundMain == 0이면 false 0이 아니면 True
    if (foundMain)
    {
        //ttempNode.type 은 1
        tempNode.type=1; /*integer*/

        //자른 문자 다음부터 구분자 또 찾기
        firstword=strtok(NULL," ");

        //tempNode.exp_data의 exp_data = firstword[0]
        tempNode.exp_data=firstword[0];


        //자른 문자 다음부터 구분자  찾기
        firstword=strtok(NULL," ");

        //(2)-1
        /* check for = */
        //firstword가 '='이라면 true
        if (!strcmpi("=",firstword))
        {
            //firstword는 자른 문자 다음부터 구분자  찾기은 값이다
            firstword=strtok(NULL," ");
        }

        //문자열을 정수 타입 변화 ex) 1
        //tempNode의 val = firstword를 정수 변환
        tempNode.val=atoi(firstword);
        //tempNode의 line은  0
        tempNode.line=0;
        //STACK은 Push(tempNode,STACK)한 값
        STACK=Push(tempNode,STACK);
    }
}
```

firstword=int

firstword=c

firstword= “=“

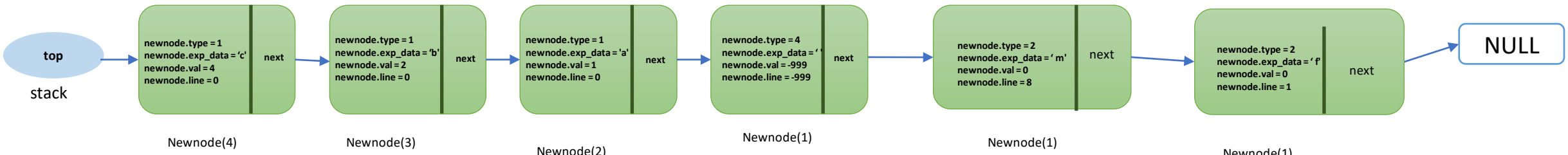
tempNode.val = 4

Push

```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```

Node자료형의newnode포인터선언;
만약 newnode는 Node타입크기 만큼메모리를 할당받으며,
newnode가NULL일 경우 ERROR, Couldn't allocate memory... 출력
NULL리턴

newnode.type = 1
newnode.val = 4
newnode.exp_data = ‘c’
newnode.line = 0
newnode 의 next 는 stck의 top이 된다.
stck의 top은 newnode이다
반환값 stck



```
//파일 스트림의 끝을 만날 때까지 반복
while (!feof(filePtr))
{

    int k=0;

    //입력 스트림에서 문자열 읽기

    fgets(line,4096, filePtr); /* read the file by Line by Line */
    /* scan for /t characters. get rid of them! */
    while(line[k]!='₩0')
    {
    if (line[k]=='₩t')
    {
       line[k]=' ';
    }

    k++;
    }

    strcpy(lineyedek,line);

    curLine++;
    tempNode.val=-999;
    tempNode.exp_data=' ';
    tempNode.line=-999;
    tempNode.type=-999;

    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }
    else{ ... }
```

curLine=13

input1.sql(읽을 파일)

```
function f(int a)
begin
   int b = 6;
   int c = 2;
   ((b+c)/a);
end

function main()
begin
   int a = 1;
   int b = 2;
   int c = 4;
   ((6 + f(c) ) / b);
end
```

1. 파일 스트림의 끝을 만날 때까지 반복
2. 정수 k의 값은 0
3. fget을 통해 ((6 + f(c)) / b);을 읽어 line에 저장 .
4. whil문 line[k]이 null 이 아닐 시때까지 반복.
   4-1) 만약 line[k]가 \t(탭키)일 시, line[k]는 ' '으로 바꾼다.
   k증가
5. line을 lineyedek에 문자열 복사
6. curLine을 1증가

tempNode.val= -999; // tempNode.val값을 -999로 대입
tempNode.exp_data=' '; // tempNode.exp_data값을 ' '로 대입
tempNode.line=-999;// tempNode.line 을 -999로 대입
tempNode.type=-999;// tempNode.type 을 -999로 대입

(6 + f(c)) / b);

firstword[0]= (라면

```
else if (firstword[0]=='(')
{

//foundMain == 0이면 false 0이 아니면 True
if (foundMain)
    {

    int i=0;
    int y=0; //3

    //OpStack + MathStack
    //MathStack.top은 NULL
    MathStack->top=NULL;
    /* now make the postfix calculcation */



    //(3)-2
    //lineyedek[i]이 NULL이 아닐때까지 반복
    while(lineyedek[i]!='₩x0')
    {
        //(3)-3-1
        /* evulate the function */
        //숫자라면 true
        if (isdigit(lineyedek[i])){ ... }
         /* ... */

        //(3)-3-2
        //lineyedek[i]이 ')'이라면
        else if (lineyedek[i]==')'){ ... }

        ////(3)-3-3
        //lineyedek[i]이 '+'이거나 lineyedek[i]이 '-' 이거나 lineyedek[i]이 '+' 이거나 lineyedek[i]이 '/'이라면
        else if ((lineyedek[i]=='+') | (lineyedek[i]=='-') | (lineyedek[i]=='+') | (lineyedek[i]=='/')){ ... }

        ////(3)-3-4
        //알파벳 대문자 "A-Z"는 1를 반환,알파벳 소문자 'a-z"는 2를 반환.
        //lineyedek[i] 영어라면
        else if (isalpha(lineyedek[i])>0){ ... }

        //1 증가
        i++;
    }//while문
```

```
/* evulate the function */
//숫자라면 true
if (isdigit(lineyedek[i])) {
    postfix[y]=lineyedek[i];
    y++;

}
 /*      */
```

postfix[0] = 6

(6 + f(c)) / b);

만약 isStackEmpty( ) 0이아니라면

```
////(3)-3-3
//lineyedek[i]이 '+'이거나 lineyedek[i]이 '-' 이거나 lineyedek[i]이 '+' 이거나 lineyedek[i]이 '/'이라면
else if ((lineyedek[i]=='+') | (lineyedek[i]=='-') | (lineyedek[i]=='*') | (lineyedek[i]=='/'))
{
   ////(3)-3-3-1
   /*operators*/
   //00이면 false 10이면 true
   //처음일 때
   if (isStackEmpty(MathStack) != 0 ) //1일때 - stck->top==0
   {
/* if stack empty push the operator to stack */
      //+
      MathStack=PushOp(lineyedek[i],MathStack);
   }
   ////(3)-3-3-2

   //0일때 - stck -top = 00이 아닐때
   //처음이 아닐때
   else { ... }
}
```

ⓐ

ⓑ

PushOp(+, MathStack)

isStackEmpty

```
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)

1. 만약 stck의 top가 0이라면
   1. 반환값 1
2. 아니면 반환값 0

PushOp

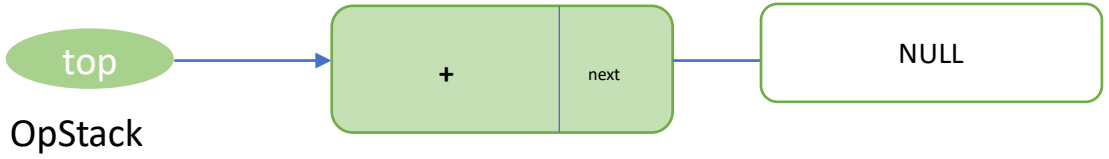
```
OpStack * PushOp(char op,OpStack *opstck)
{
opNode *newnode;
if ((newnode=(opNode*)malloc(sizeof(opNode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->op=op;
newnode->next=opstck->top;
opstck->top=newnode;
return opstck;
}
}
```

OpStack * PushOp(char op,OpStack *opstck)

1. opNode자료형의 newnode포인터선언;
2. 만약 newnode는 opNode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   반환NULL
3. 아니라면,
   1. newnode의 op는 매개변수+이다.
   2. newnode의 next는 opstck의 top이다.
   3. opstck의 top은 newnode이다.

반환값 opstck

PushOp



*** input1.sql는이부분 실행x

input1.sql(읽을 파일)

```
function f(int a)
begin
   int b = 6;
   int c = 2;
   ((b+c)/a);
end

function main()
begin
   int a = 1;
   int b = 2;
   int c = 4;
   ((6 + f(c) ) / b);
end
```

만약 isStackEmpty( ) 0이라면

```
////(3)-3-3
//lineyedek[i]이 '+'이거나 lineyedek[i]이 '-' 이거나 lineyedek[i]이 '+' 이거나 lineyedek[i]이 '/'이라면
else if ((lineyedek[i]=='+') | (lineyedek[i]=='-') | (lineyedek[i]=='*') | (lineyedek[i]=='/'))
{
   ////(3)-3-3-1
   /*operators*/
   //00이면 false 10이면 true
   //처음일 때
   if (isStackEmpty(MathStack) != 0 ) //1일때 - stck->top==0
   {
/* if stack empty push the operator to stack */
      //+
      MathStack=PushOp(lineyedek[i],MathStack);
   }
   ////(3)-3-3-2

   //0일때 - stck -top = 00이 아닐때
   //처음이 아닐때
   else { ... }
}
```

```
// ...
else
{
   //////(3)-3-3-2->1
   /* check for presedence */
   //현재 +,- <= 이전(/,*)
   if (Priotry(lineyedek[i]) <= Priotry(MathStack->top->op) )
   {
      /* higher presedence for example + < *  */
      /* pop the last operator */

      /* add the poped operator to the postfix */
      //postfix[y]은 PopOp(MathStack)반환값
      postfix[y]=PopOp(MathStack);
      //y값 1증가
      y++;

      //MathStack은 PushOp(lineyedek[i],MathStack)이다
      MathStack=PushOp(lineyedek[i],MathStack);


   }
   //////(3)-3-3-2->2
   else
   {
      /* lower presedence for example / > + */
      /* push to stack */
      //MathStack은 PushOp(lineyedek[i],MathStack)의 반환값
      MathStack=PushOp(lineyedek[i],MathStack);
   }

}
```

*** input1.sql는이부분 실행x
만약 isStackEmpty( ) 0이라면

```
// ...
else
{
    //////(3)-3-3-2->1
    /* check for presedence */
    //현재 +,- <= 이전(/,*)
    if (Priotry(lineyedek[i]) <= Priotry(MathStack->top->op) )
    {
        /* higher presedence for example + < *  */
        /* pop the last operator */

        /* add the poped operator to the postfix
        //postfix[y]은 PopOp(MathStack)반환값
        postfix[y]=PopOp(MathStack);
        //y값 1증가
        y++;

        //MathStack은 PushOp(lineyedek[i],MathStack)이다
        MathStack=PushOp(lineyedek[i],MathStack);


    }
    //////(3)-3-3-2->2
    else
    {
        /* lower presedence for example / > + */
        /* push to stack */
        //MathStack은 PushOp(lineyedek[i],MathStack)의 반환값
        MathStack=PushOp(lineyedek[i],MathStack);
    }

}
```

Priotry

```
int Priotry(char operator)
{
  if ((operator=='+') | (operator=='-'))
    return 1;
  else if ((operator=='/') | (operator=='*'))
    return 2;
return 0;
}
```

int Priotry(char operator)

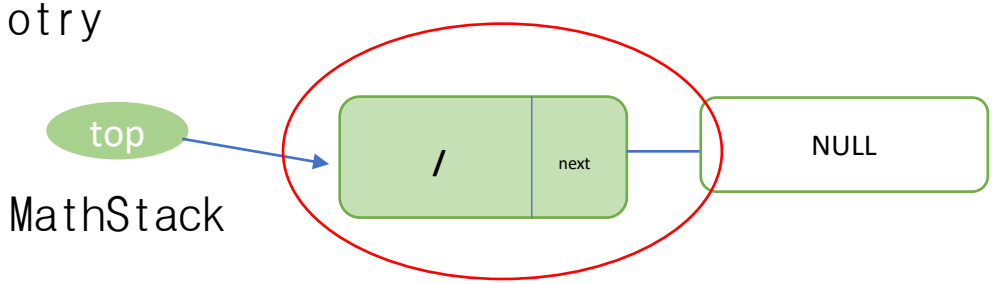
1. 만약 operator 가 '+'이거나 operato가 '-'라면
    1. 반환값 1
2. 만약 operator가 '/'이거나 operator가 '*'이라면
    1. 반환값 2
3. 반환값 0

Priotry(MathStack->top->op)의 반환값이 Priotry(lineyedek[i])의 반환값보다 크거나 같을때

예) Priotry (lineyedek[i])
lineyedek[i] == '+' 일 때, **반환값 1**

Priotry (MathStack->top->op])은 /일때, **반환값 2**

Priotry



*** input1.sql는이부분 실행x

현재값보다 MathStack->top->op이 크거나 값다면

```
// ...
else
{
    //////(3)-3-3-2->1
    /* check for presedence */
    //현재 +,- <= 이전(/,*)
    if (Priotry(lineyedek[i]) <= Priotry(MathStack->top->op) )
    {
        /* higher presedence for example + < *  */
        /* pop the last operator */

        /* add the poped operator to the postfix */
        //postfix[y]은 PopOp(MathStack)반환값
        postfix[y]=PopOp(MathStack);
        //y값 1증가
        y++;

        //MathStack은 PushOp(lineyedek[i],MathStack)이다
        MathStack=PushOp(lineyedek[i],MathStack);


    }
    //////(3)-3-3-2->2
    else
    {
        /* lower presedence for example / > + */
        /* push to stack */
        //MathStack은 PushOp(lineyedek[i],MathStack)의 반환값
        MathStack=PushOp(lineyedek[i],MathStack);
    }

}
}
```

PopOp

```
char PopOp(OpStack *opstck)
{
opNode *temp;
char op;
if (opstck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
op=opstck->top->op;
temp=opstck->top;
opstck->top=opstck->top->next;
free(temp);
return op;
}
return NULL;
}
```

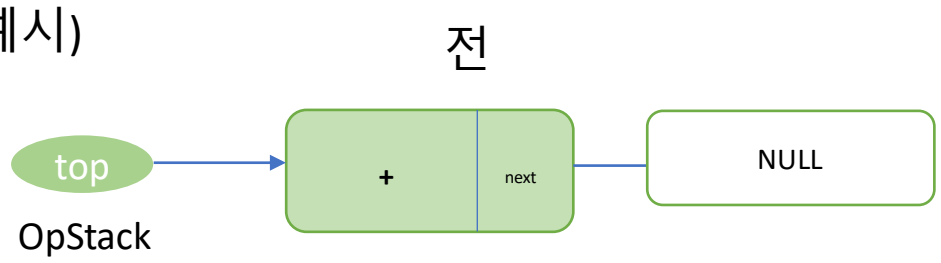
char PopOp(OpStack *opstck)

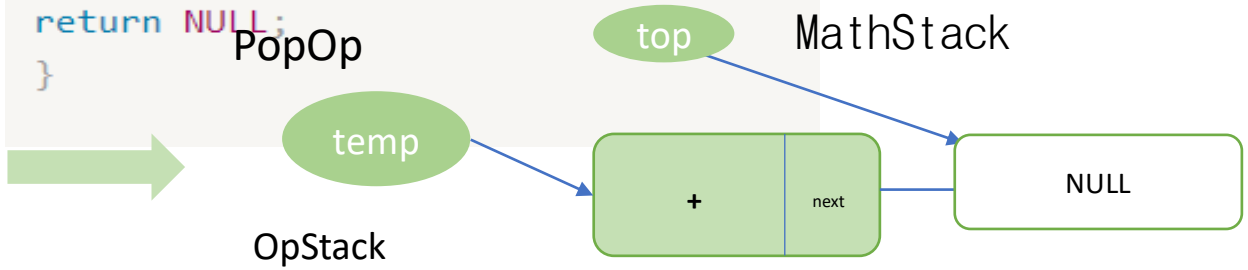
1. opNode구조체의 자료형 temp 포인터 선언
2. 문자자료형 op선언
3. 만약 opstck의 top이 NULL일 경우
   printf("ERROR, empty stack..."); 출력

4. 아니라면,
   1. op는 opstck의 top의 op이다
   2. temp 는 opstck의 top이다.
   3. opstck의 top은 opstck의 top의 next이다.
   temp 동적메모리 해제
   반환값 op

반환값 NULL

예시)





***코드에 이부분 실행x

```
// ...
else
{
    //////(3)-3-3-2->1
    /* check for presedence */
    //현재 +,- <= 이전(/,*)
    if (Priotry(lineyedek[i]) <= Priotry(MathStack->top->op) )
    {
        /* higher presedence for example + < *  */
        /* pop the last operator */

        /* add the poped operator to the postfix */
        //postfix[y]은 PopOp(MathStack)반환값
        postfix[y]=PopOp(MathStack);
        //y값 1증가
        y++;

        //MathStack은 PushOp(lineyedek[i],MathStack)이다
        MathStack=PushOp(lineyedek[i],MathStack);


    }
    //////(3)-3-3-2->2
    else
    {
        /* lower presedence for example / > + */
        /* push to stack */
        //MathStack은 PushOp(lineyedek[i],MathStack)의 반환값
        MathStack=PushOp(lineyedek[i],MathStack);
    }

}
}
```

PushOp

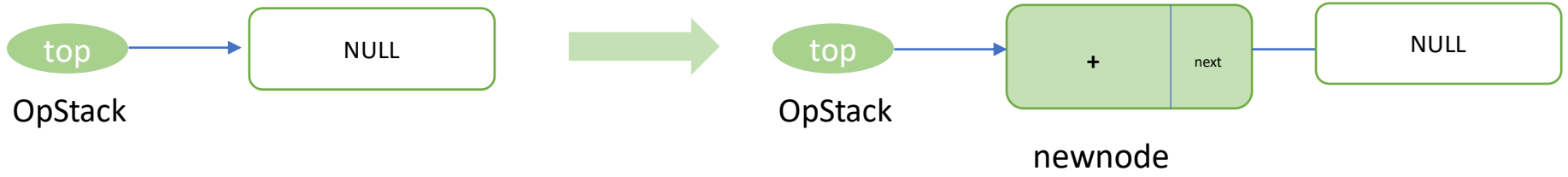
```
OpStack * PushOp(char op,OpStack *opstck)
{
opNode *newnode;
if ((newnode=(opNode*)malloc(sizeof(opNode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->op=op;
newnode->next=opstck->top;
opstck->top=newnode;
return opstck;
}
}
```

OpStack * PushOp(char op,OpStack *opstck)

1. opNode자료형의 newnode포인터선언;
2. 만약 newnode는 opNode 타입크기 만큼메모리를 할당받으며, newnode가 NULL 경우
   ERROR, Couldn't allocate memory... 출력
   반환NULL
3. 아니라면,
   1. newnode의 op는 매개변수op이다.
   2. newnode의 next는 opstck의 top이다.
   3. opstck의 top은 newnode이다.

반환값 opstck

예시(



***코드에 이부분 실행x

```
// ...
else
{
    //////(3)-3-3-2->1
    /* check for presedence */
    //현재 +,- <= 이전(/,*)
    if (Priotry(lineyedek[i]) <= Priotry(MathStack->top->op) )
    {
        /* higher presedence for example + < *  */
        /* pop the last operator */

        /* add the poped operator to the postfix */
        //postfix[y]은 PopOp(MathStack)반환값
        postfix[y]=PopOp(MathStack);
        //y값 1증가
        y++;

        //MathStack은 PushOp(lineyedek[i],MathStack)이다
        MathStack=PushOp(lineyedek[i],MathStack);


    }
    //////(3)-3-3-2->2
    else
    {
        /* lower presedence for example / > + */
        /* push to stack */
        //MathStack은 PushOp(lineyedek[i],MathStack)의 반환값
        MathStack=PushOp(lineyedek[i],MathStack);
    }
}
```

현재값보다 MathStack->top->op이 크거나 값지 않다면

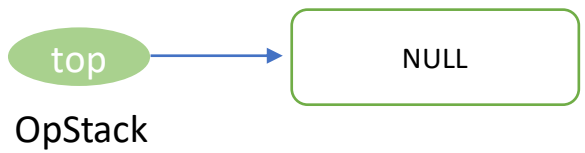
```
OpStack * PushOp(char op,OpStack *opstck)
{
opNode *newnode;
if ((newnode=(opNode*)malloc(sizeof(opNode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->op=op;
newnode->next=opstck->top;
opstck->top=newnode;
return opstck;
}
}
```

OpStack * PushOp(char op,OpStack *opstck)

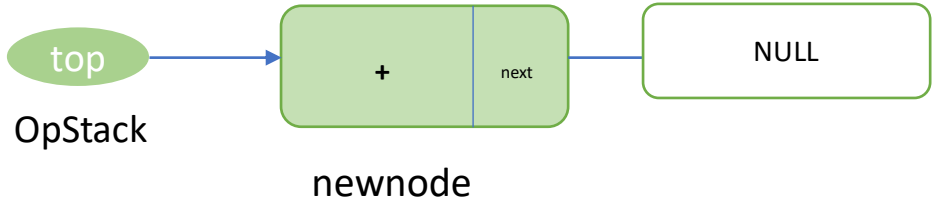
1. opNode자료형의 newnode포인터선언;
2. 만약 newnode는 opNode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   반환NULL
3. 아니라면,
   1. newnode의 op는 매개변수op이다.
   2. newnode의 next는 opstck의 top이다.
   3. opstck의 top은 newnode이다.

반환값 opstck

예시(



PushOp



(6 + f(c)) / b);

```
//lineyedek[i] 영어라면
else if (isalpha(lineyedek[i])>0)
{
    int codeline=0;
    int dummyint=0;
    /*look if it's a variable or function call
    int retVal=0;
    retVal=GetVal(lineyedek[i],&codeline,STACK);
    //int codeline =1

    //-1


    ////(3)-3-4-1
    if ((retVal!=-1) & (retVal!=-999)){ ... }
    ////(3)-3-4-2
    //-1
    else{ ... }
}
```

GetVal (f,&codeline,STACK)

GetVal

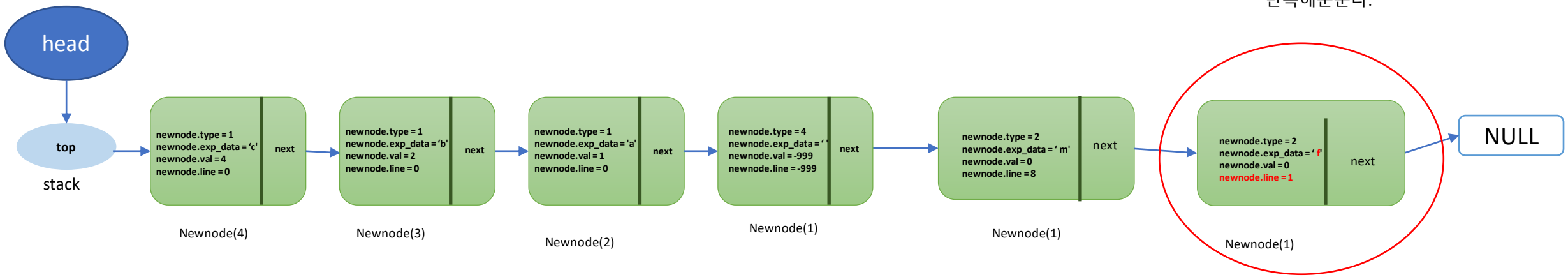
```
int GetVal(char exp_name,int * line,Stack *stck)
{

Node * head;
*line=0;
    if (stck->top == NULL )
    {
        printf("ERROR, empty stack...");
    }
    else
    {
        head=stck->top;
        do
        {
            if ( head->exp_data==exp_name )
            {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }

            }
            else
            {
                head=head->next;
            }
        } while (head->next!=NULL);
        /* check again once more */
        if (head->exp_data==exp_name)
        {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }


        }
    }

    return -999;
}
```

head->exp_data = f

*line = 1
return -1

GetVal()
자료형인 Node인 포인트 head선언

1. head를 stck의 top으로 둔다.
2. head의exp_data== exp_name
3. 와 같다면
    1. head의 type이 1이라면 head의 val이 반환값이다.
    2. head의 type이 2이라면
       line은head의 line이다.
       -1 반환값
4. head의exp_data==exp_name
5. 와 같은게 없다면 head는 head의 next를 해준다.
   head의 next가 NULL이 아닐때까지
   반복해준준다.



(6 + f(c)) / b);

```
if (LastFunctionReturn==-999)
{

/* if function */
/* add to our system stack that we are making a call to function */
int j;
tempNode.type=3;
tempNode.line=curLine;
STACK=Push(tempNode,STACK);

/* get function's arguments value */
CalingFunctionArgVal=GetVal(lineyedek[i+2],&dummyint,STACK);

fclose(filePtr);
filePtr=fopen(argv[1],"r");
curLine=0;
/* file reversed to start postion */
/* now go codeline lines to go, to the functions line */


codeline olabilir */
for(j=1;j<codeline;j++)
{
   fgets(dummy,4096,filePtr); /*
   curLine++;
}

WillBreak=1;
break;
}
```

CalingFunctionArgVal =4 (c값)

File을 닫고 읽기모드로 다시연다

curLine 은 0이다

Dummy에 저장

WillBreak=1

push

```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```

Push(sNode,*stck)
새로운 노드의 타입은 3이되며,
새로운노드의 val은 0,새로운 노드의
exp_data = ‘ ‘이며,

```
int GetVal(char exp_name,int * line,Stack *stck)
{

Node * head;
*line=0;
    if (stck->top == NULL )
    {
        printf("ERROR, empty stack...");
    }
    else
    {
        head=stck->top;
        do
        {
            if ( head->exp_data==exp_name )
            {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }
```

head->exp_data = c

if(head->exp_data = c)이라면
head의 type이 1이라면
head의 val 리턴

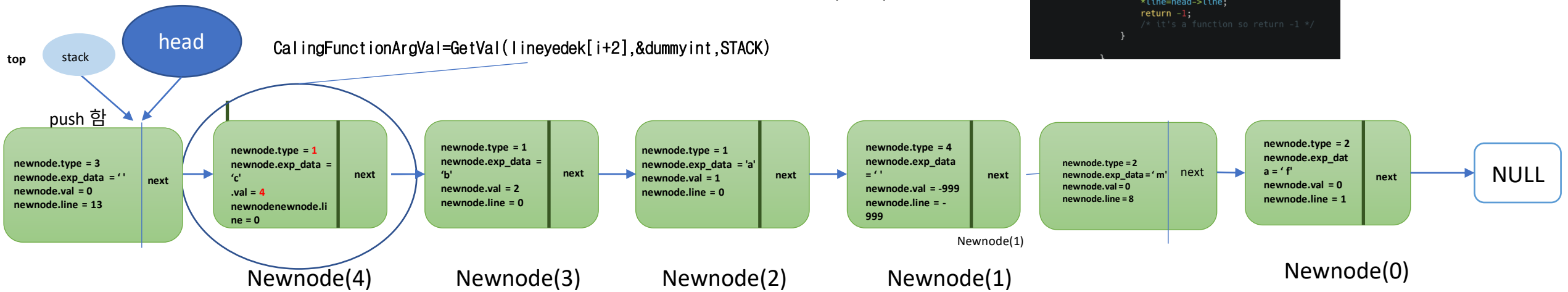
top

stack

head

CalingFunctionArgVal=GetVal(lineyedek[i+2],&dummyint,STACK)

push 함



```
    if (LastFunctionReturn==-999)
    {
    /* if function */
    /* add to our system stack that we are making a call to function */
    int j;
    tempNode.type=3;
    tempNode.line=curLine;
    STACK=Push(tempNode,STACK);

    /* get function's arguments value */
    CalingFunctionArgVal=GetVal(lineyedek[i+2],&dummyint,STACK);



    fclose(filePtr);
    filePtr=fopen(argv[1],"r");
    curLine=0;
    /* file reversed to start postion */
    /* now go codeline lines to go, to the functions line */


codeline olabilir */
    for(j=1;j<codeline;j++)
    {
        fgets(dummy,4096,filePtr); /* read the file by Line by Line */
        curLine++;
    }

    WillBreak=1;
    break;
    }
```

break를 만나 whil문을
빠져나간다.

```
//lineyedek[i]이 NULL이 아닐때까지 반복
while(lineyedek[i]!='₩x0')
{
    //(3)-3-1
    /* evulate the function */
    //숫자라면 true
    if (isdigit(lineyedek[i])) {
        postfix[y]=lineyedek[i];

        y++;  //1증가

    }
     /* ... */

    //(3)-3-2
    //lineyedek[i]이 ')'이라면
    else if (lineyedek[i]==')')
    {
        //(3)-3-2-1
        ////0이면 true 1이면 false
        //0일때
        if (!isStackEmpty(MathStack) != 0 ) //- OpStack->top==0이 아닐 때
        {
            postfix[y]=PopOp(MathStack);
            y++; // 1증가
        }
    }

    ////(3)-3-3
    //lineyedek[i]이 '+'이거나 lineyedek[i]이 '-' 이거나 lineyedek[i]이 '*' 이거나 lineyedek[i]이 '/'이라면
    else if ((lineyedek[i]=='+') | (lineyedek[i]=='-') | (lineyedek[i]=='*') | (lineyedek[i]=='/')){ ... }

    ////(3)-3-4
    //알파벳 대문자 'A-Z'는 1을 반환.알파벳 소문자 'a-z'는 2를 반환.
    //lineyedek[i] 영어라면
    else if (isalpha(lineyedek[i])>0){ ... }

    //1 증가
    i++;
}//while문



//WillBreak이 0이라면
if (WillBreak==0){ ... }
WillBreak=0;
}
```

다시 reset

function f(int a)

```
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //fget을 통해 function f(int a)을 읽어 line에 저장 .
    fgets(line,4096,filePtr); /* read the file by Line by Line */
    /* scan for /t characters, get rid of them! */

    // whil문 line[k]이  null 이 아닐 시때까지 반복.
    while(line[k]!='₩0')
    {
       //  4-1) 만약 line[k]가 ₩t(탭키)일 시,
     if (line[k]=='₩t')
     {
       //line[k]는  ' '으로 바꾼다.
      line[k]=' ';
     }
      //k에 1증가
     k++;
    }
    //5. line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을  ' '로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //begin과 line이 동일하다면
    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }


    //ⓑ
    //  //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //line이 end라면
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }

    //ⓒ begin과 end가 아니면
    else{ ... }
```

curLine 은 1이다

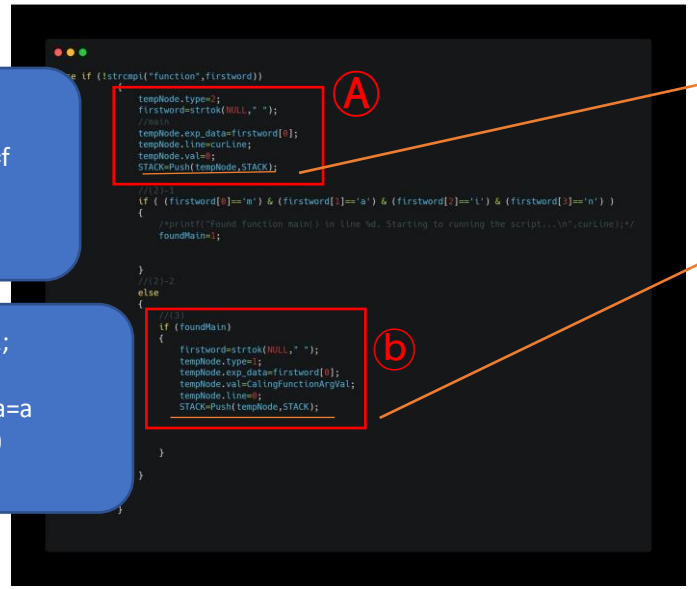
다시 reset

function f(int a)

tempNode.type=2;
firstword= f(int
tempNode.exp_data=f
tempNode.line=1
tempNode.val=0;

tempNode.type=1;
firstword= a)
tempNode.exp_data=a
tempNode.line=0
tempNode.val=4;

function

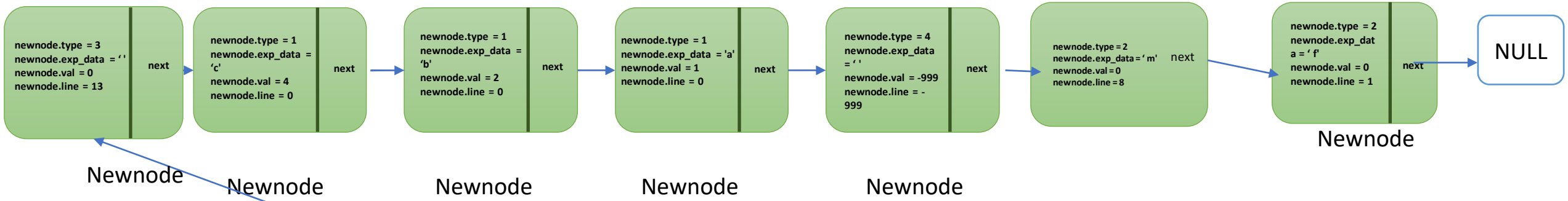


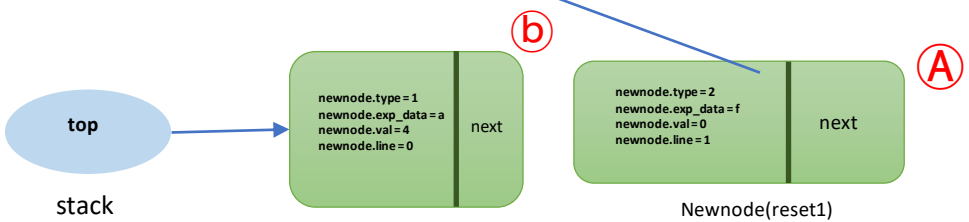
Push

```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```

Node자료형의newnode포인터선언;
만약 newnode는 Node타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우 ERROR, Couldn't allocate memory... 출력
NULL리턴

newnode.type = 1
newnode.val = 4
newnode.exp_data = ' a'
newnode.line = 0
newnode 의 next 는 stck의 top이 된다.
stck의 top은 newnode이다
반환값 stck





파일을 다시 새로 시작한다.

파일반복

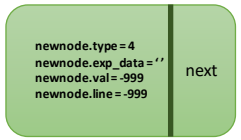
```
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //fget을 통해 function f(int a)을 읽어 line에 저장 .
    fgets(line,4096,filePtr); /* read the file by Line by Line */
    /* scan for /t characters. get rid of them! */

    // whil문 line[k]이  null 이 아닐 시때까지 반복.
    while(line[k]!='₩0')
    {
       //  4-1) 만약 line[k]가 ₩t(탭키)일 시,
     if (line[k]=='₩t')
     {
       //line[k]는  ' '으로 바꾼다.
      line[k]=' ';
     }
      //k에 1증가
     k++;
    }
    //5. line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을  ' '로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  둘이 동일하면 1(true) 반환하고 동일하지 않으면 0(false)
    //begin과 line이 동일하다면
    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }


    //ⓑ
    //  //strcmpi :  둘이 동일하면 1(true) 반환하고 동일하지 않으면 0(false)
    //line이 end라면
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }

    //ⓒ begin과 end가 아니면
    else{ ... }
```

input1.sql(읽을 파일)

```
function f(int a)
begin
   int b = 6;
   int c = 2;
   ((b+c)/a);
end

function main()
begin
   int a = 1;
   int b = 2;
   int c = 4;
   ((6 + f(c) ) / b);
end
```

begin

begin이라면



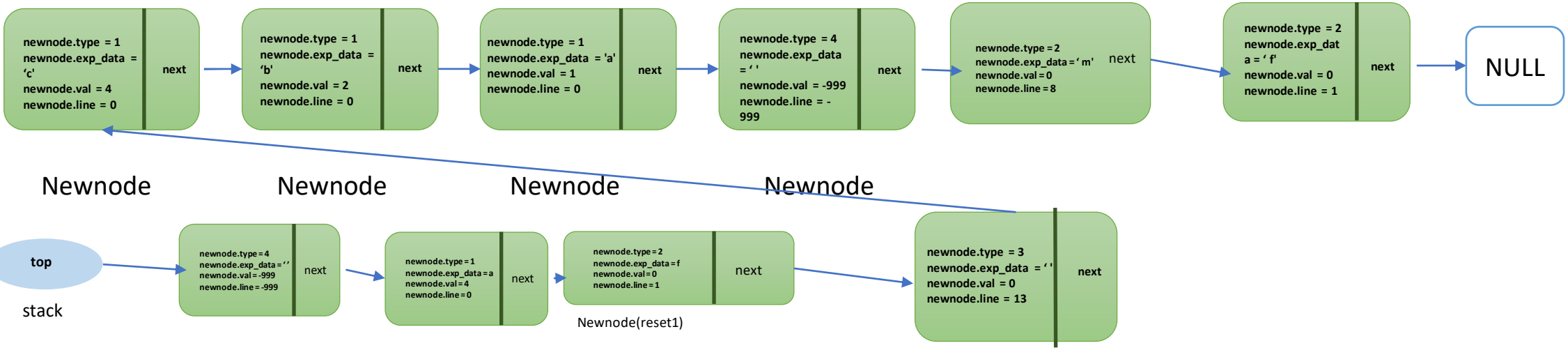

```
//begind이면
if (!strcmpi("begin\n",line) | !strcmpi("begin",line))
{
    //foundMain == 0이면 false 0이 아니면 True
    if (foundMain)
    {
        tempNode.type=4;
        STACK=Push(tempNode,STACK);
    }
}
```

Push

```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```

Node자료형의newnode포인터선언;
만약 newnode는 Node타입크기 만큼메모리를 할당받으며,
newnode가NULL일 경우 ERROR, Couldn't allocate memory... 출력
NULL리턴

newnode.type = 4
newnode.val = -999
newnode.exp_data = ‘’
newnode.line = -999
newnode 의 next 는 stck의 top이 된다.
stck의 top은 newnode이다
반환값 stck



파일반복

int b = 6;

```
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //fget을 통해 function f(int a)을 읽어 line에 저장 .
    fgets(line,4096,filePtr); /* read the file by Line by Line */
    /* scan for /t characters. get rid of them! */

    // whil문 line[k]이  null 이 아닐 시때까지 반복.
    while(line[k]!='₩0')
    {
       //  4-1) 만약 line[k]가 ₩t(탭키)일 시,
     if (line[k]=='₩t')
     {
       //line[k]는  ' '으로 바꾼다.
      line[k]=' ';
     }
      //k에 1증가
     k++;
    }
    //5. line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을  ' '로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  둘이 동일하면 1(true) 반환하고 동일하지 않으면 0(false)
    //begin과 line이 동일하다면
    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }


    //ⓑ
    //  //strcmpi :  둘이 동일하면 1(true) 반환하고 동일하지 않으면 0(false)
    //line이 end라면
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ) { ... }

    //ⓒ begin과 end가 아니면
    else{ ... }
```

input1.sql(읽을 파일)

```
function f(int a)
begin
   int b = 6;
   int c = 2;
   ((b+c)/a);
end

function main()
begin
   int a = 1;
   int b = 2;
   int c = 4;
   ((6 + f(c) ) / b);
end
```

int b = 6;

```
if (!strcmpi("int",firstword))
        {
            //(1)-1
            //foundMain ==0
            //foundMain == 0이면 false 0이 아니면 True
            if (foundMain)
            {
                tempNode.type=1; /*integer*/

                //자른 문자 다음부터 구분자 또 찾기
                firstword=strtok(NULL," ");

                tempNode.exp_data=firstword[0];


                //자른 문자 다음부터 구분자 또 찾기
                firstword=strtok(NULL," ");

                //(2)-1
                /* check for = */
                if (!strcmpi("=",firstword))
                {
                    //다음값을 탐색
                    firstword=strtok(NULL," ");
                }

                //문자열을 정수 타입 변화 ex) 1
                tempNode.val=atoi(firstword);
                tempNode.line=0;
                STACK=Push(tempNode,STACK);
            }
        }
```

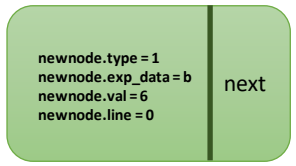
```
//ttempNode.type 은 1
tempNode.type=1; /*integer*/

//자른 문자 다음부터 구분자 또 찾기
firstword=strtok(NULL," ");

//tempNode.exp_data의 exp_data = firstword[0]
tempNode.exp_data=b;


//자른 문자 다음부터 구분자  찾기
firstword=strtok(NULL," "); // =

//(2)-1
/* check for = */
//firstword가 '='이라면 true
if (!strcmpi("=",firstword))
{
//firstword는 자른 문자 다음부터 구분자  찾기은
값이다.
firstword=strtok(NULL," ");
}

//문자열을 정수 타입 변화 ex) 1
//tempNode의 val = firstword를 정수 변환
tempNode.val=6
//tempNode의 line은  0
tempNode.line=0;
//STACK은 Push(tempNode,STACK)한 값
STACK=Push(tempNode,STACK);
```

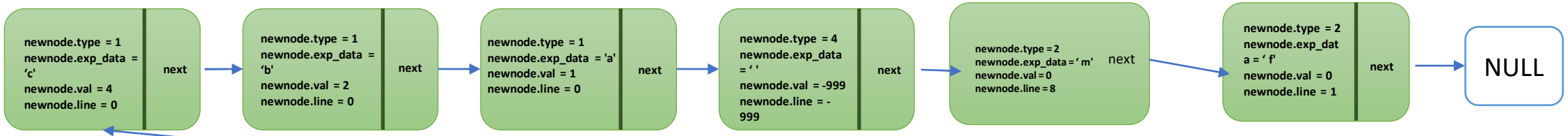
Push

```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```

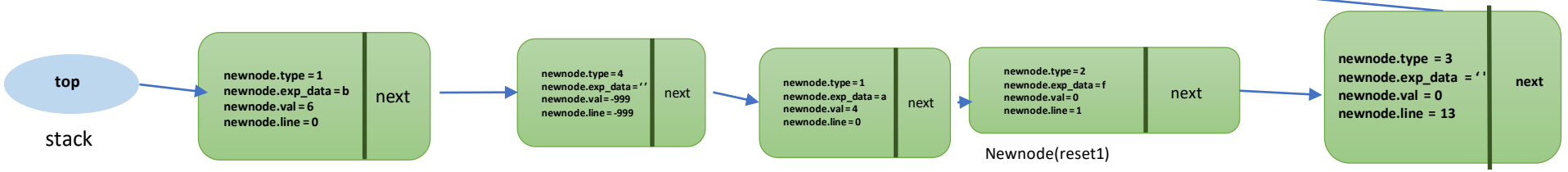
Node자료형의newnode포인터선언;
만약 newnode는 Node타입크기 만큼메모리를 할당받으며,
newnode가 NULL일 경우 ERROR, Couldn't allocate memory... 출력
NULL리턴

newnode.type = 1
newnode.val = -6
newnode.exp_data = b
newnode.line = 0
newnode 의 next 는 stck의 top이 된다.
stck의 top은 newnode이다
반환값 stck





Newnode Newnode Newnode Newnode



int c = 2;

파일반복

```
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //fget을 통해 function f(int a)을 읽어 line에 저장 .
    fgets(line,4096,filePtr); /* read the file by Line by Line */
    /* scan for /t characters, get rid of them! */

    // whil문 line[k]이  null 이 아닐 시때까지 반복.
    while(line[k]!='₩0')
    {
        //  4-1) 만약 line[k]가 ₩t(탭키)일 시,
      if (line[k]=='₩t')
      {
        //line[k]는  ' '으로 바꾼다.
       line[k]=' ';
      }
      //k에 1증가
     k++;
    }
    //5. line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을  ' '로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  둘이 동일하면 1(true) 반환하고 동일하지 않으면 0(false)
    //begin과 line이 동일하다면
    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }


    //ⓑ
    //  //strcmpi :  둘이 동일하면 1(true) 반환하고 동일하지 않으면 0(false)
    //line이 end라면
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }

    //ⓒ begin과 end가 아니면
    else{ ... }
```

input1.sql(읽을 파일)

```
function f(int a)
begin
    int b = 6;
    int c = 2;
    ((b+c)/a);
end

function main()
begin
    int a = 1;
    int b = 2;
    int c = 4;
    ((6 + f(c) ) / b);
end
```

int c = 2;

```
if (!strcmpi("int",firstword))
        {
            //(1)-1
            //foundMain ==0
            //foundMain == 0이면 false 0이 아니면 True
            if (foundMain)
            {
                tempNode.type=1; /*integer*/

                //자른 문자 다음부터 구분자 또 찾기
                firstword=strtok(NULL," ");

                tempNode.exp_data=firstword[0];


                //자른 문자 다음부터 구분자 또 찾기
                firstword=strtok(NULL," ");

                //(2)-1
                /* check for = */
                if (!strcmpi("=",firstword))
                {
                    //다음값을 탐색
                    firstword=strtok(NULL," ");
                }

                //문자열을 정수 타입 변화 ex) 1
                tempNode.val=atoi(firstword);
                tempNode.line=0;
                STACK=Push(tempNode,STACK);
            }
        }
```

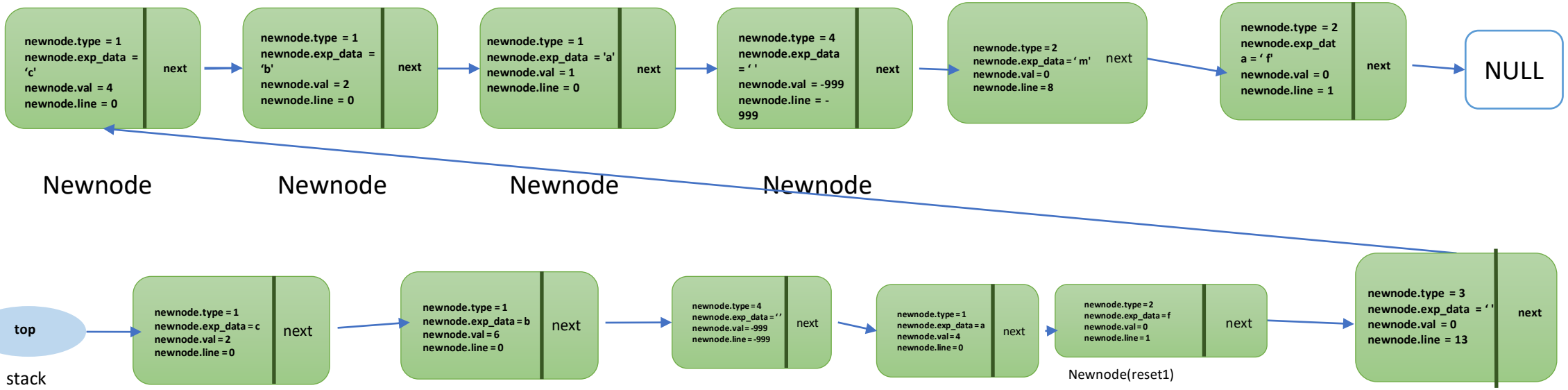
```
//ttempNode.type 은 1
tempNode.type=1; /*integer*/

//자른 문자 다음부터 구분자 또 찾기
firstword=strtok(NULL," ");

//tempNode.exp_data의 exp_data =
firstword[0]
tempNode.exp_data=c;


//자른 문자 다음부터 구분자  찾기
firstword=strtok(NULL," "); // =

//(2)-1
/* check for = */
//firstword가 '='이라면 true
if (!strcmpi("=",firstword))
{
//firstword는 자른 문자 다음부터 구분자
찾기은 값이다.
firstword=strtok(NULL," ");
}

//문자열을 정수 타입 변화 ex) 1
//tempNode의 val = firstword를 정수 변환
tempNode.val=2
//tempNode의 line은  0
tempNode.line=0;
//STACK은 Push(tempNode,STACK)한 값
STACK=Push(tempNode,STACK);
```

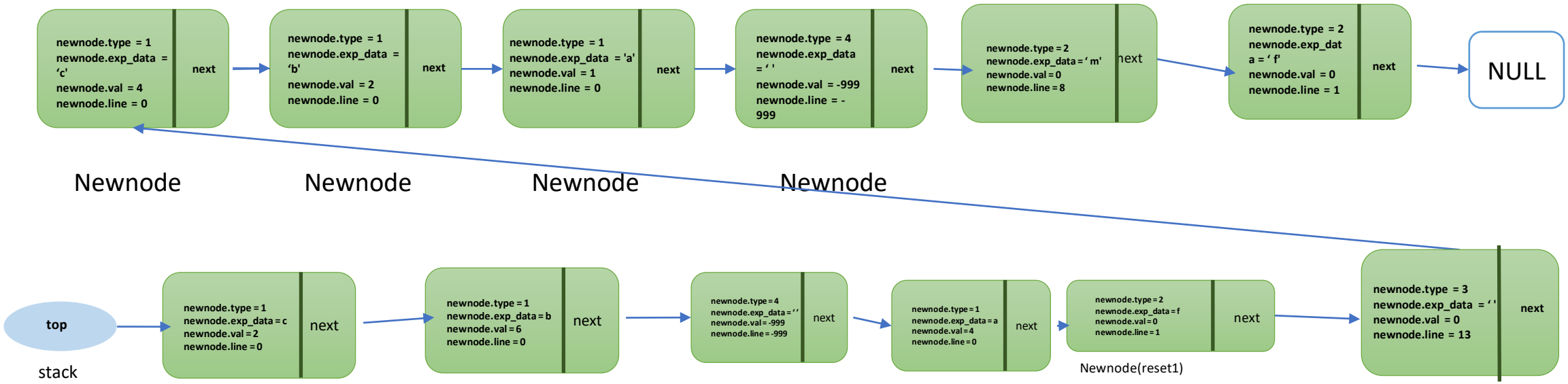
Push

```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```

Node자료형의newnode포인터선언;
만약 newnode는 Node타입크기 만큼메모리를 할당받으며,
newnode가 NULL일 경우 ERROR, Couldn't allocate memory... 출력
NULL리턴

newnode.type = 1
newnode.val = 2
newnode.exp_data = c
newnode.line = 0
newnode 의 next 는 stck의 top이 된다.
stck의 top은 newnode이다
반환값 stck




newnode.type = 1
newnode.exp_data = 'c'
newnode.val = 4
newnode.line = 0
next
newnode.type = 1
newnode.exp_data = 'b'
newnode.val = 2
newnode.line = 0
next
newnode.type = 1
newnode.exp_data = 'a'
newnode.val = 1
newnode.line = 0
next
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
newnode.type = 2
newnode.exp_data = ' m'
newnode.val = 0
newnode.line = 8
next
newnode.type = 2
newnode.exp_data = ' f'
newnode.val = 0
newnode.line = 1
next
NULL
Newnode
Newnode
Newnode
Newnode
top
stack
newnode.type = 1
newnode.exp_data = c
newnode.val = 2
newnode.line = 0
next
newnode.type = 1
newnode.exp_data = b
newnode.val = 6
newnode.line = 0
next
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
newnode.type = 1
newnode.exp_data = a
newnode.val = 4
newnode.line = 0
next
newnode.type = 2
newnode.exp_data = f
newnode.val = 0
newnode.line = 1
next
Newnode(reset1)
newnode.type = 3
newnode.exp_data = ' '
newnode.val = 0
newnode.line = 13
next

(b+c)/a);

세분화2-4-1-1 (begin과 end가 아니면) - - '('라면 세분화 ⓐ

```
//3초기화
else if (firstword[0]=='(')
{
//(3)-1
if (foundMain)
    {

    int i=0;
    int y=0;

    //OpStack * MathStack
    MathStack->top=NULL;
    /* now make the postfix calcu
```

MathStack ->top = NULL로 설정

(b+c)/a);

세분화2-4-1-1 (begin과 end가 아니면) - '('라면 세분화 ⓐ

알파벳이라면

```
else if (isalpha(lineyedek[i])>0)
                {
                    int codeline=0;
                    int dummyint=0;
                    /*look if it's a variable or function call
                    int retVal=0;
                    retVal=GetVal(lineyedek[i],&codeline,STACK);
```

retVal = 6

①

GetVal()

```
int GetVal(char exp_name,int * line,Stack *stck)
{
Node * head;
*line=0;
    if (stck->top == NULL )
    {
        printf("ERROR, empty stack...");
    }
    else
    {
        head=stck->top;
        do
        {
            if ( head->exp_data==exp_name )
            {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }

            }
            else
            {
                head=head->next;
            }
        } while (head->next!=NULL);
        /* check again once more */
        if (head->exp_data==exp_name)
        {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }


        }
    }
    return -999;
}
```

②

```
////(3)-3-4-1
if ((retVal!=-1) & (retVal!=-999))
{
    /* if variable */
    postfix[y]=retVal+48; /* in asci
    y++; //1

}
```

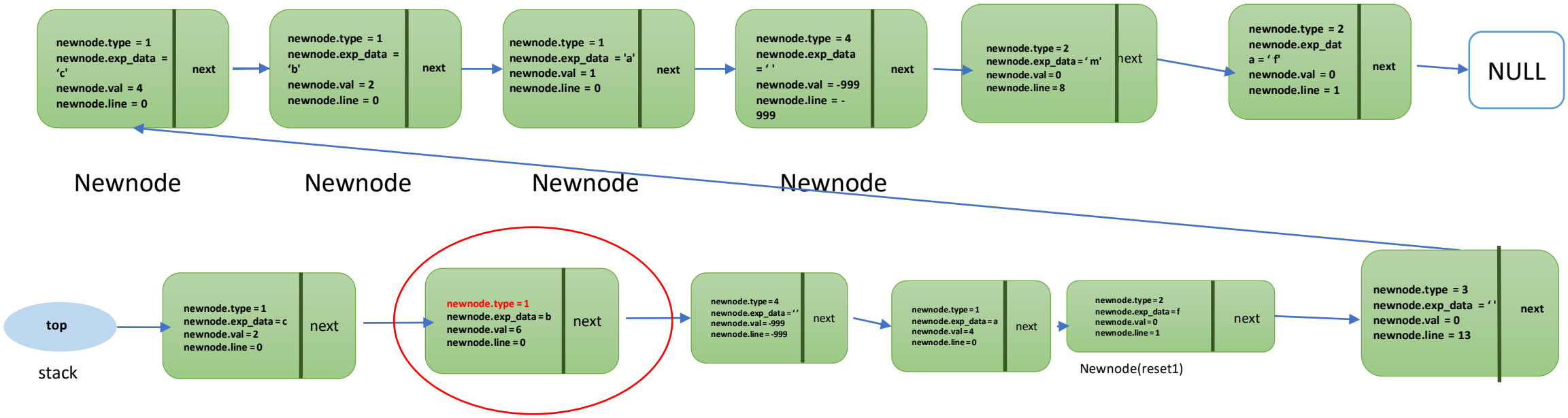
retVal은 -1이 아니며 -999가 아니기에
postfix[1] = 6+48;을 한 후
Y값에 1을 더한다.

Postfix[0] =54

GetVal()
1.Head를 stck의 top이 된다.
2.head의 exp_data가 b 가 아니라면, head->next가 NULL이 아닐때까지 head = head->next 한다
4.만약 head의 exp_data가 b 이고 head의type이 1이라면 6이 리턴된다.

((b+c)/a);

GetVal()


newnode.type = 1
newnode.exp_data = 'c'
newnode.val = 4
newnode.line = 0
next
newnode.type = 1
newnode.exp_data = 'b'
newnode.val = 2
newnode.line = 0
next
newnode.type = 1
newnode.exp_data = 'a'
newnode.val = 1
newnode.line = 0
next
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
newnode.type = 2
newnode.exp_data = ' m'
newnode.val = 0
newnode.line = 8
next
newnode.type = 2
newnode.exp_data = ' f'
newnode.val = 0
newnode.line = 1
next
NULL
Newnode
Newnode
Newnode
Newnode
top
stack
newnode.type = 1
newnode.exp_data = c
newnode.val = 2
newnode.line = 0
next
newnode.type = 1
newnode.exp_data = b
newnode.val = 6
newnode.line = 0
next
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
newnode.type = 1
newnode.exp_data = a
newnode.val = 4
newnode.line = 0
next
newnode.type = 2
newnode.exp_data = f
newnode.val = 0
newnode.line = 1
next
Newnode(reset1)
newnode.type = 3
newnode.exp_data = ' '
newnode.val = 0
newnode.line = 13
next

((b+c)/a);

MathStack->top=NULL;

```
//0이면 false 1이면 true
//처음일 때
if (isStackEmpty(MathStack) != 0 ) //1일때 - stck->top==0
{
/* if stack empty push the operator to stack */
   //+
   MathStack=PushOp(lineyedek[i],MathStack);
}
```

+,MathStack

isStackEmpty

```
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)

1. 만약 stck의 top가 0이라면
   1. 반환값 1
2. 아니면 반환값 0

PushOp

```
OpStack * PushOp(char op,OpStack *opstck)
{
opNode *newnode;
if ((newnode=(opNode*)malloc(sizeof(opNode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->op=op;
newnode->next=opstck->top;
opstck->top=newnode;
return opstck;
}
}
```

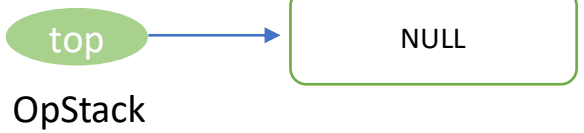
PushOp()

만약 opNode의 구조체 크기가 Null일 경우, "ERROR, Couldn't allocate memory..."출력 후 NULL 리턴

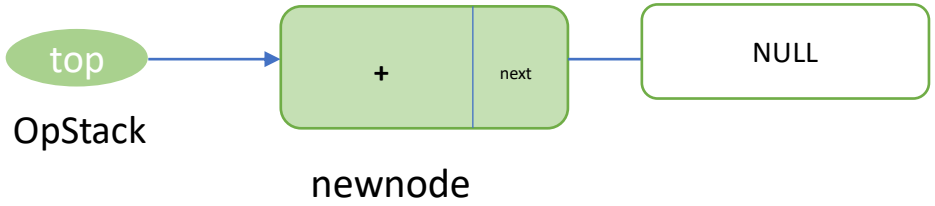
아니라면,
1. +는 새로운 노드의 op가 된다.
2. opstck->top은 새로운 노드의 next이다
3. newnode는 Opstck의 top은 가리킨다.

return opstck;

MathStack



PushOp



((b+c)/a);

```
else if (isalpha(lineyedek[i])>0)
                {
                    int codeline=0;
                    int dummyint=0;
                    /*look if it's a va
                    int retVal=0;
                    retVal=GetVal(lineyedek[i],&codeline,STACK);
```

c,0,stack

retVal= 2

GetVal

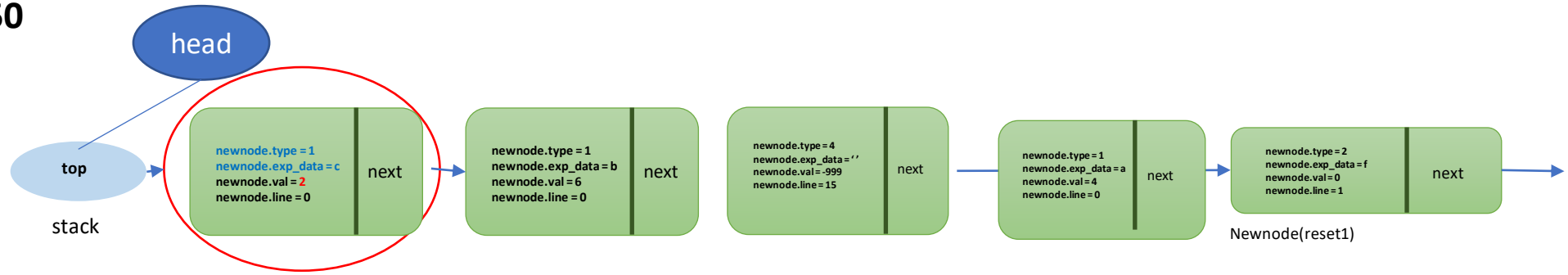
```
head=stck->top;
do
{
    if ( head->exp_data==exp_name )
    {

        if (head->type==1)
        {
            /* return the variables value */
            return head->val;
        }
```

1.Head를 stck의 top이 된다.
2.head의 exp_data가 c가 아니라면, head->next가 NULL이 아닐때까지 head = head->next 한다
4.만약 head의 exp_data가 c 이고 head의type이 1이라면head.val 인2이 리턴된다.

```
////(3)-3-4-1
if ((retVal!=-1) & (retVal!=-999))
{
    /* if variable */
    postfix[y]=retVal+48; /* in asci
    y++; //1
}
```

만약 retVal이 -1이 아니거나 -999가 아니라면,
Postfix[] = retVal+48을한 후, y 1증가

Postfix[1] =50



((b+c)/a);

세분화2-4-1-1 (begin과 end가 아니면) - '('라면 세분화 ⓐ

')'이라면

```
else if (lineyedek[i]==')')
                {
                    //(3)-3-2-1
                    ////0이면 true 1이면 false

                    if (!isStackEmpty(MathStack) != 0 ) //0일때 - OpStack-
>top==0이 아닐 때
                    {
                        // y = 0
                        //Null과 0은 같다
                        postfix[y]=PopOp(MathStack); //
                        y++;
                    }
                }
```

Postfix[2] ='+'

Postfix[]

54	50	'+'

isStackEmpty

```
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)

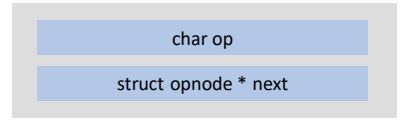
1. 만약 stck의 top가 0이라면
   1. 반환값 1
2. 아니면 반환값 0

PopOp(opstck)

1. opstck의 top이 NULL이면 "Error, empty stack…" 라고 하고 return null;

opstck의 top이 NULL이 아니면,

1. op =+ (opstck의 top의 op)를 가리킨다.
2. temp는 opstck의 top이다.
3. opstck의 top은opstck의 top의 next이다.
4. temp를 메모리 해체를한다.
5. return은 +

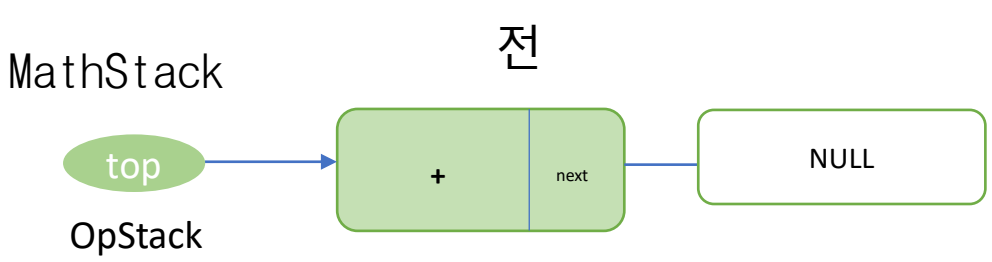


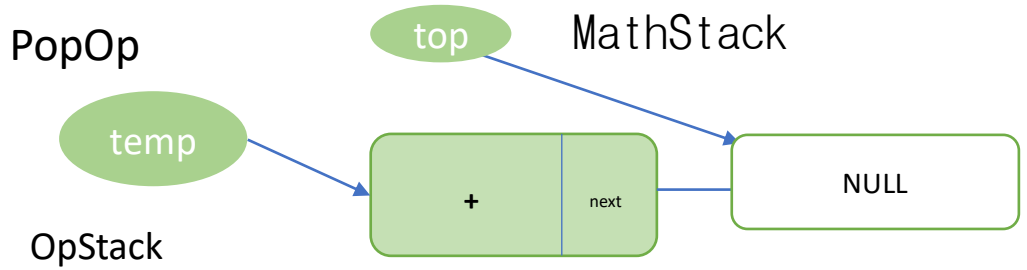
opNode

```
char PopOp(OpStack *opstck)
{
opNode *temp;
char op;
if (opstck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
op=opstck->top->op;
temp=opstck->top;
opstck->top=opstck->top->next;
free(temp);
return op;
}
return NULL;
}
```

Op = +

메모리 해제





((b+c)/a);

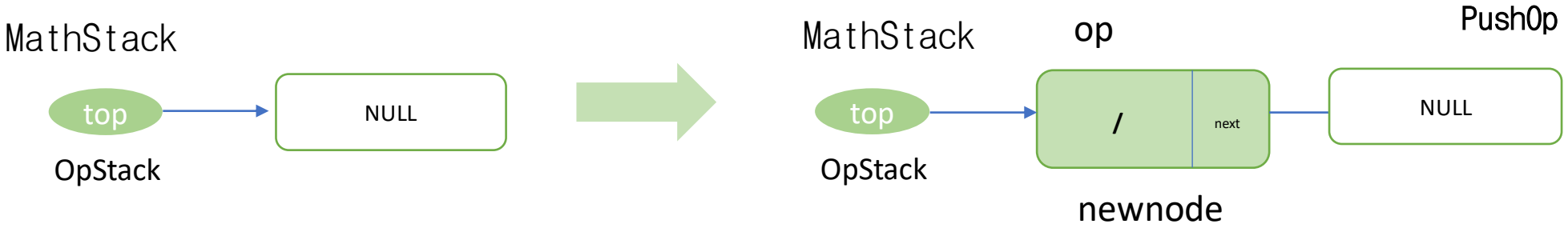
```
    else if ((lineyedek[i]=='+') | (lineyedek[i]=='-') | (lineyedek[i]=='*') |
(lineyedek[i]=='/'))
                {
                    ////(3)-3-3-1
                    /*operators*/
                    //0이면 false 1이면 true
                    //처음일 때
                    if (isStackEmpty(MathStack) != 0 ) //1일때 - stck->top==0
                    {
                /* if stack empty push the operator to stack */

                        MathStack=PushOp(lineyedek[i],MathStack);
                    }
                    ////(3)-3-
```

PsushOp('/' , MathStack)

```
OpStack * PushOp(char op,OpStack *opstck)
{
opNode *newnode;
if ((newnode=(opNode*)malloc(sizeof(opNode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->op=op;
newnode->next=opstck->top;
opstck->top=newnode;
return opstck;
}
}
```

PushOp()
만약 opNode의 구조체 크기가 Null일 경우,
"ERROR, Couldn't allocate memory..."출력 후 NULL 리턴

아니라면,
1. /는 새로운 노드의 op가 된다.
2. opstck->top은 새로운 노드의 next이다
3. newnode는 Opstck의 top은 가리킨다.
return opstck;



((b+c)/a);

```
else if (isalpha(lineyedek[i])>0)
                {
                    int codeline=0;
                    int dummyint=0;
                    /*look if it's a v
                    int retVal=0;
                    retVal=GetVal(lineyedek[i],&codeline,STACK);
```

a,0,stack

①

②

retVal = 4

```
////(3)-3-4-1
if ((retVal!=-1) & (retVal!=-999))
{
    /* if variable */
    postfix[y]=retVal+48; /* in asci
    y++; //1

}
```

retVal은 -1이 아니며 -999가 아니기에
postfix[3] = 4+48;을 한 후
Y값에 1을 더한다.

GetVal

```
head=stck->top;
do
{
    if ( head->exp_data==exp_name )
    {

        if (head->type==1)
        {
            /* return the variables value
            return head->val;
        }
        else if (head->type==2)
        {
            *line=head->line;
            return -1;
            /* it's a function so return -1 */
        }

    }
    else
    {
        head=head->next;
    }
} while (head->next!=NULL);
/* check agin once more */
```

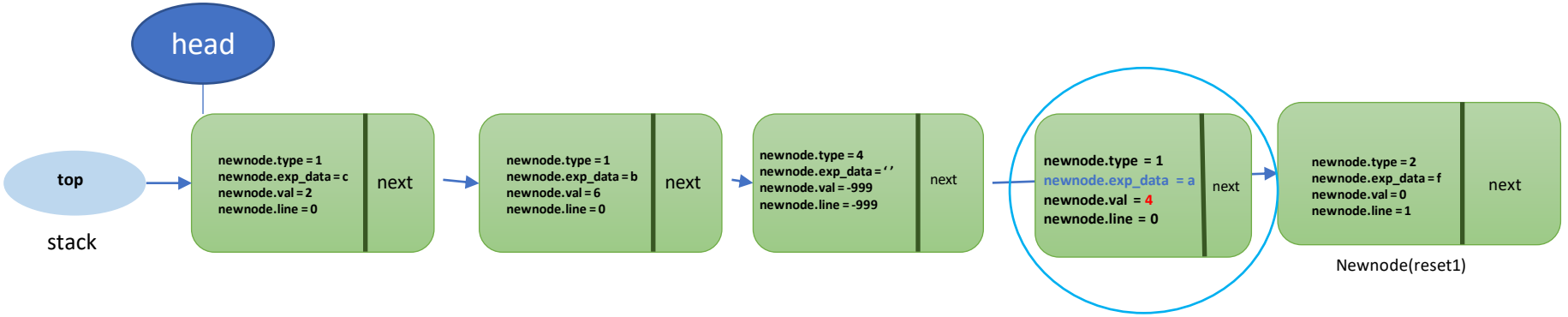
Return 4

GetVal()
1.Head를 stck의 top이 된다.
2.head의 exp_data가 a 가 아니라면, head->next가 NULL이 아닐때까지 head = head->next 한다
4.만약 head의 exp_data가 a 이고 head의type이 1이라면 head의 val 즉 4가 리턴된다.

Postfix[3] =4+48

Postfix[]

54	50	'+'	52



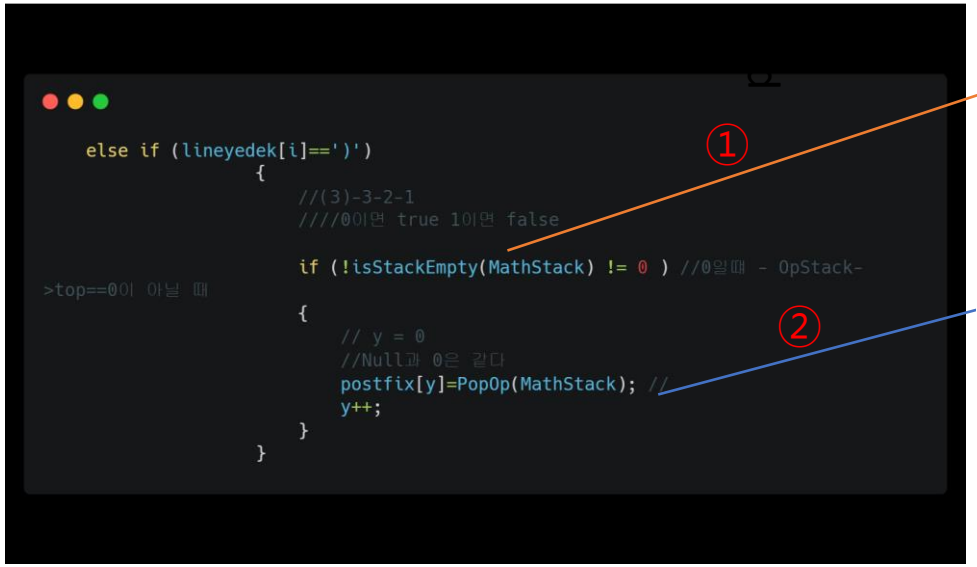
((b+c)/a);

int isStackEmpty(OpStack *stck)
1. 만약 stck의 top가 0이라면
   1. 반환값 1
2. 아니면 반환값 0

isStackEmpty

```
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

Stck-> top에 값이 이있기에 0 리턴



PopOp(opstck)
1. opNode구조체의 자료형 temp 포인터 선언
2. opstck의 top이 NULL이면 “Error, empty stack…” 라고 하고 return null;

opstck의 top이 NULL이 아니면,
1. op =/ (opstck의 top의 op)를 가리킨다.
2. temp는 opstck의 top이다.
3. opstck의 top은opstck의 top의 next이다.
4. temp를 메모리 해체를한다.
5. return은 /

```
char PopOp(OpStack *opstck)
{
opNode *temp;
char op;
if (opstck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
op=opstck->top->op;
temp=opstck->top;
opstck->top=opstck->top->next;
free(temp);
return op;
}
return NULL;
}
```

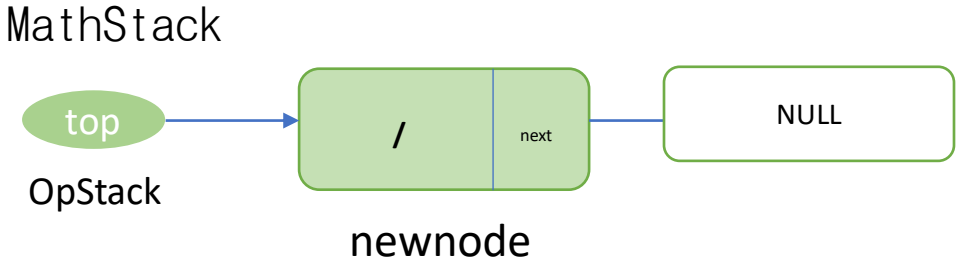
메모리 해제

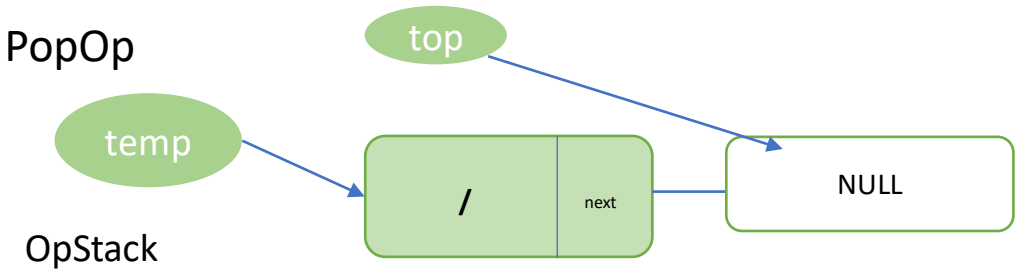
Op = /

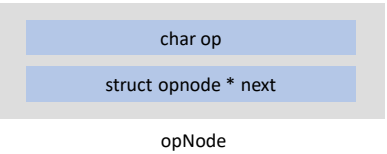
Postfix[4] =‘/’

Postfix[]

54	50	'+'	52	'/'







```c
if (WillBreak==0)
{
/* get out items left in the mathstack */
while (isStackEmpty(MathStack)==0)
{

    /* add the poped operator to the postfix */
    postfix[y]=PopOp(MathStack);
    y++;
}

postfix[y]='\0';

//MathStack=FreeAll(MathStack);

/* now calculate the postfix */
/*printf("\nCURRENT POSTFIX=%s\n",postfix);*/

i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
        val1=PopPostfix(CalcStack);

        val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;


}
```

①

②

isStackEmpty

```c
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)

1. 만약 stck의 top가 0이라면
    1. 반환값 1
2. 아니면 반환값 0

```c
char PopOp(OpStack *opstck)
{
opNode *temp;
char op;
if (opstck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
op=opstck->top->op;
temp=opstck->top;
opstck->top=opstck->top->next;
free(temp);
return op;
}
return NULL;
}
```

char PopOp(OpStack *opstck)

1. opNode구조체의 자료형 temp 포인터 선언
2. 문자자료형 op선언
3. 만약 opstck의 top이 NULL일 경우
   printf("ERROR, empty stack..."); 출력
4. 아니라면,
   1. op는 opstck의 top의 op이다
   2. temp 는 opstck의 top이다.
   3. opstck의 top은 opstck의 top의 next이다.
   temp 동적메모리 해제
   반환값 op

반환값 NULL

```
if (WillBreak==0)
{
/* get out items left in the mathstack */
while (isStackEmpty(MathStack)==0)
{

    /* add the poped operator to the postfix */
    postfix[y]=PopOp(MathStack);
    y++;
}

postfix[y]='₩0';

//MathStack=FreeAll(MathStack);

/* now calculate the postfix */
/*printf("₩nCURRENT POSTFIX=%s₩n",postfix);*/

i=0;

CalcStack->top=NULL;
```

①

현재 MathStack
OpStack은 NULL이기에 while문 실행x

②

isStackEmpty

```
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)

1. 만약 stck의 top가 0이라면
   1. 반환값 1
2. 아니면 반환값 0

```
char PopOp(OpStack *opstck)
{
opNode *temp;
char op;
if (opstck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
op=opstck->top->op;
temp=opstck->top;
opstck->top=opstck->top->next;
free(temp);
return op;
}
return NULL;
}
```

char PopOp(OpStack *opstck)

1. opNode구조체의 자료형 temp 포인터 선언
2. 문자자료형 op선언
3. 만약 opstck의 top이 NULL일 경우
   printf("ERROR, empty stack..."); 출력
4. 아니라면,
   1. op는 opstck의 top의 op이다
   2. temp 는 opstck의 top이다.
   3. opstck의 top은 opstck의 top의 next이다.
   temp 동적메모리 해제
   반환값 op

반환값 NULL

만약 Opstack stck의 top이 0이 아니라면
Opstack에 있는 값들을 PopOp함수를 통해 op값을
postfix[] 에 Opstack stck 의 top이 0 될때까지
반복하여 저장한다.

ⓐ `postfix[y]='\0';`

```
//MathStack=FreeAll(MathStack);

/* now calculate the postfix */
/*printf("\nCURRENT POSTFIX=%s\n",postfix);*/

i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
        val1=PopPostfix(CalcStack);

        val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

Postfix[i]이 \0일때까지 반복실행

Postfix[i]가 숫자일시,

'0'은 아스키코드 48

1증가

Postfix[]

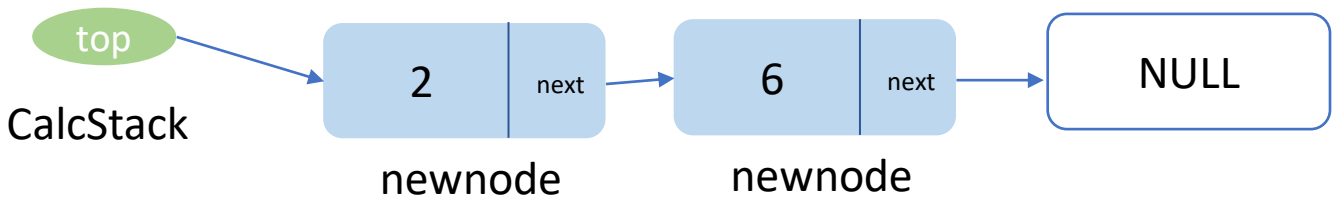
ⓐ

54	50	'+'	52	'/'	\0

```
PostfixStack * PushPostfix(int val,PostfixStack *poststck)
{
Postfixnode *newnode;
if ((newnode=(Postfixnode*)malloc(sizeof(Postfixnode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->val=val;
newnode->next=poststck->top;
poststck->top=newnode;
return poststck;
}
}
```

PostfixStack * PushPostfix(int val,PostfixStack *poststck)

1. Postfixnode구조체의 자료형인 newnode 포인터 선언
2. 만약 newnode는 Postfixnode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   NULL반환
3. 아니면라면,
   1. newnode의 val 은 매개변수 val이다
   2. newnode의 next 는 poststck의 top이다.
   3. poststck의 top 는 newnode이다.
   4. 반환값 poststck



```
i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
①       val1=PopPostfix(CalcStack);

②       val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

val1 = 2
val2=6

resultVal = 6+2

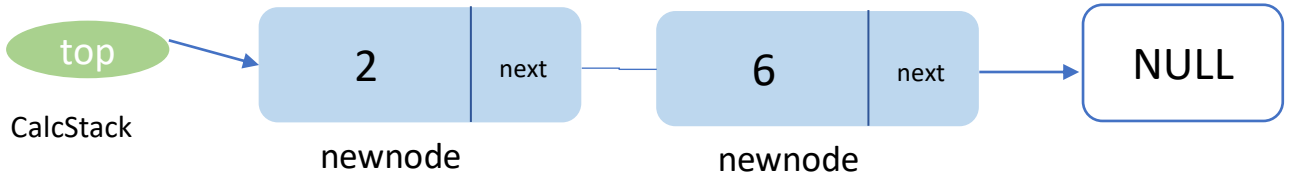
Postfix[]

54	50	'+'	52	'/'	\0

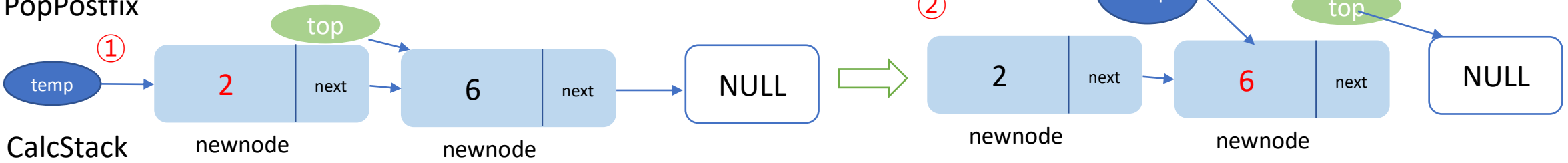
```
char PopPostfix(PostfixStack *poststck)
{
Postfixnode *temp;
int val;
if (poststck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
val=poststck->top->val;
temp=poststck->top;
poststck->top=poststck->top->next;
free(temp);
return val;
}
return NULL;
}
```

char PopPostfix(PstfixStack *poststcPostfixnode구조체의 자료k)

1. 형 temp 포인터 선언
2. 정수자료형인 val선언
3. 만약 poststck의 top이 NULL이라면
   1. ERROR, empty stack...console에 출력
4. 아니라면,
   1. val 은 poststck의 top의 val이다.
   2. tmep는 poststck의 top
   3. poststck의 top 은 poststck의 top의 next이다.
   4. temp의 동적메모리 해제
   5. 반환값 val
5. 반환값 NULL



PopPostfix



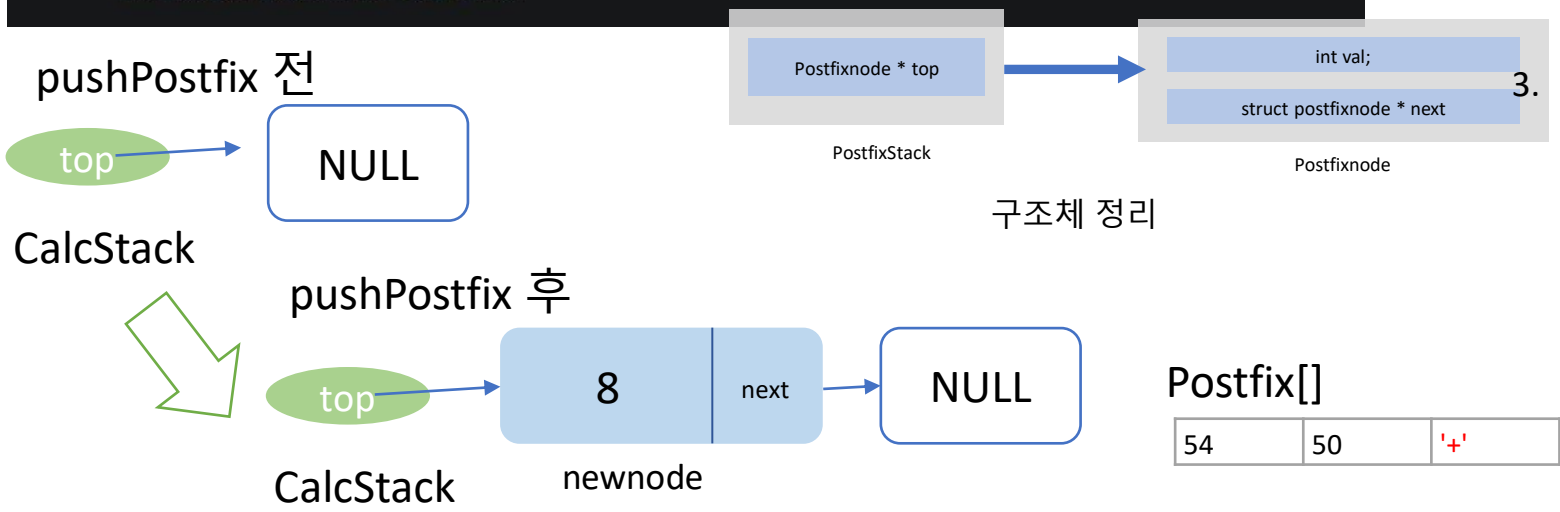
```
i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
        val1=PopPostfix(CalcStack);

        val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

pushPostfix(8,CalcStck)

```
PostfixStack * PushPostfix(int val,PostfixStack *poststck)
{
Postfixnode *newnode;
if ((newnode=(Postfixnode*)malloc(sizeof(Postfixnode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->val=val;
newnode->next=poststck->top;
poststck->top=newnode;
return poststck;
}
}
```

PostfixStack * PushPostfix(int val,PostfixStack *poststck)

1. Postfixnode구조체의 자료형인 newnode 포인터 선언
2. 만약 newnode는 Postfixnode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   NULL반환
3. 아니면라면,
   1. newnode의 val 은 매개변수 val이다
   2. newnode의 next 는 poststck의 top이다.
   3. poststck의 top 는 newnode이다.
   4. 반환값 poststck



구조체 정리

Postfix[]

54	50	'+'	52	'/'	\0

```
postfix[y]='\0';

//MathStack=FreeAll(MathStack);

/* now calculate the postfix */
/*printf("\nCURRENT POSTFIX=%s\n",postfix);*/

i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
        val1=PopPostfix(CalcStack);

        val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

Postfix[i]이 \0일때까지 반복실행

Postfix[i]가 숫자일시,

'0'은 아스키코드 48

1증가

Postfix[]

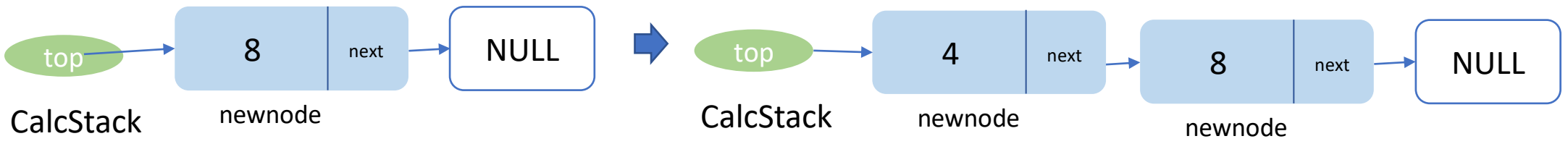
54	50	'+'	52	'/'	\0

```
PostfixStack * PushPostfix(int val,PostfixStack *poststck)
{
Postfixnode *newnode;
if ((newnode=(Postfixnode*)malloc(sizeof(Postfixnode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->val=val;
newnode->next=poststck->top;
poststck->top=newnode;
return poststck;
}
}
```

PostfixStack * PushPostfix(int val,PostfixStack *poststck)

1. Postfixnode구조체의 자료형인 newnode 포인터 선언
2. 만약 newnode는 Postfixnode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   NULL반환
3. 아니면라면,
   1. newnode의 val 은 4 이다
   2. newnode의 next 는 poststck의 top이다.
   3. poststck의 top 는 newnode이다.
   4. 반환값 poststck

PushPostfix()



```
i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
    ① val1=PopPostfix(CalcStack);

    ② val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

val1 = 4
val2=8

resultVal = 8/4

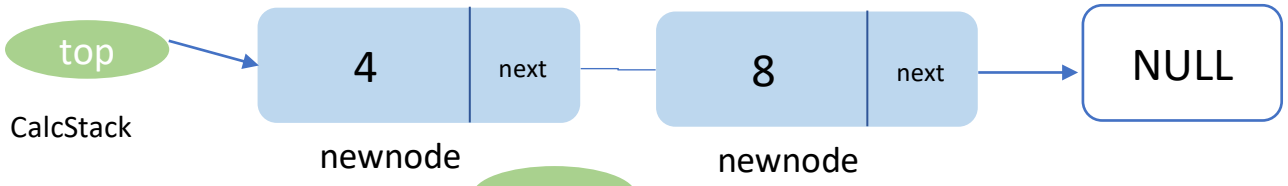
Postfix[]

54	50	'+'	52	'/'	\0

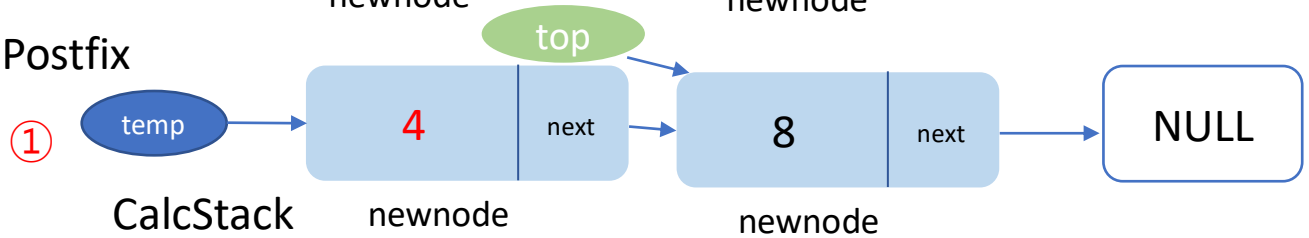
```
char PopPostfix(PostfixStack *poststck)
{
Postfixnode *temp;
int val;
if (poststck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
val=poststck->top->val;
temp=poststck->top;
poststck->top=poststck->top->next;
free(temp);
return val;
}
return NULL;
}
```

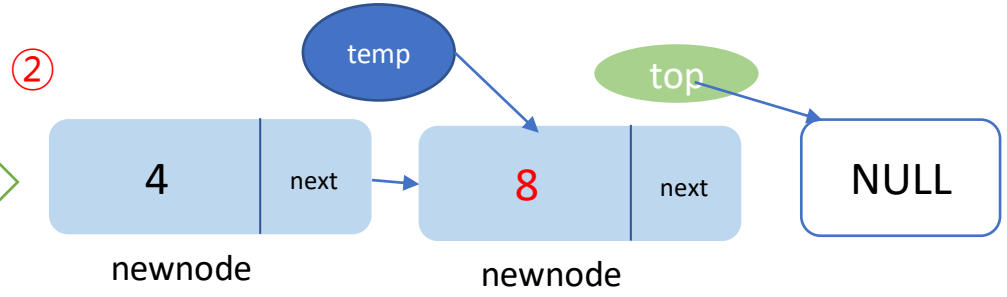
char PopPostfix(PstfixStack *poststcPostfixnode구조체의 자료k)

1. 형 temp 포인터 선언
2. 정수자료형인 val선언
3. 만약 poststck의 top이 NULL이라면
   1. ERROR, empty stack...console에 출력
4. 아니라면,
   1. val 은 poststck의 top의 val이다.
   2. tmep는 poststck의 top
   3. poststck의 top 은 poststck의 top의 next이다.
   4. temp의 동적메모리 해제
   5. 반환값 val
5. 반환값 NULL



PopPostfix





Postfix[]

54	50	'+'	52	'/'	\0

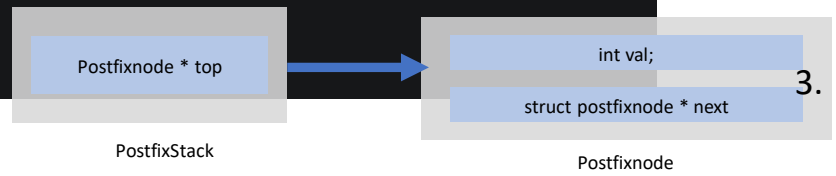
```
i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
        val1=PopPostfix(CalcStack);

        val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

pushPostfix(2,CalcStck)

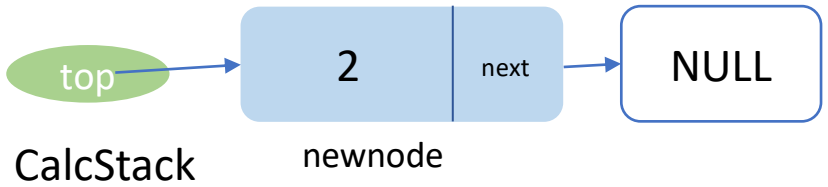
```
PostfixStack * PushPostfix(int val,PostfixStack *poststck)
{
Postfixnode *newnode;
if ((newnode=(Postfixnode*)malloc(sizeof(Postfixnode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->val=val;
newnode->next=poststck->top;
poststck->top=newnode;
return poststck;
}
}
```

PostfixStack * PushPostfix(int val,PostfixStack *poststck)

1. Postfixnode구조체의 자료형인 newnode 포인터 선언
2. 만약 newnode는 Postfixnode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   NULL반환
3. 아니면라면,
   1. newnode의 val 은 매개변수 val이다
   2. newnode의 next 는 poststck의 top이다.
   3. poststck의 top 는 newnode이다.
   4. 반환값 poststck



구조체 정리



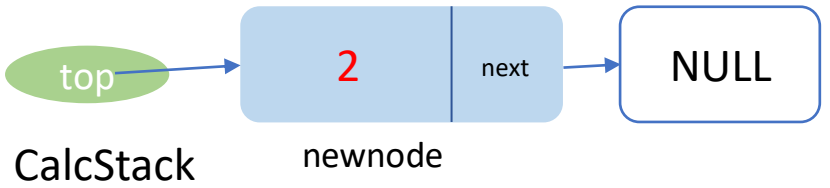
Postfix[]

54	50	'+'	52	'/'	\0

```
        postfix[y]='\0';

        //MathStack=FreeAll(MathStack);

        /* now calculate the postfix */
        /*printf("\nCURRENT POSTFIX=%s\n",postfix);*/

        i=0;

        CalcStack->top=NULL;
        while(postfix[i]!='\x0')
        {
            if (isdigit(postfix[i])) {
                /* push to stack */
                CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
            }
            else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
            {
                val1=PopPostfix(CalcStack);

                val2=PopPostfix(CalcStack);


                switch (postfix[i])
                {
                    case '+': resultVal=val2+val1;break;
                    case '-': resultVal=val2-val1;break;
                    case '/': resultVal=val2/val1;break;
                    case '*': resultVal=val2*val1;break;
                }

                CalcStack=PushPostfix(resultVal,CalcStack);
            }
            i++;
        }

        //CalcStack=FreeAll(CalcStack);
        LastExpReturn=CalcStack->top->val;
```

```
    }
    WillBreak=0;
    }
}
```

Postfix[5] = \0이기에 while반복문을 빠져나간다.



LastExpReturn= CalcStack의 top의 val이기에
LastExpReturn = 2이다.
WillBreak=0;

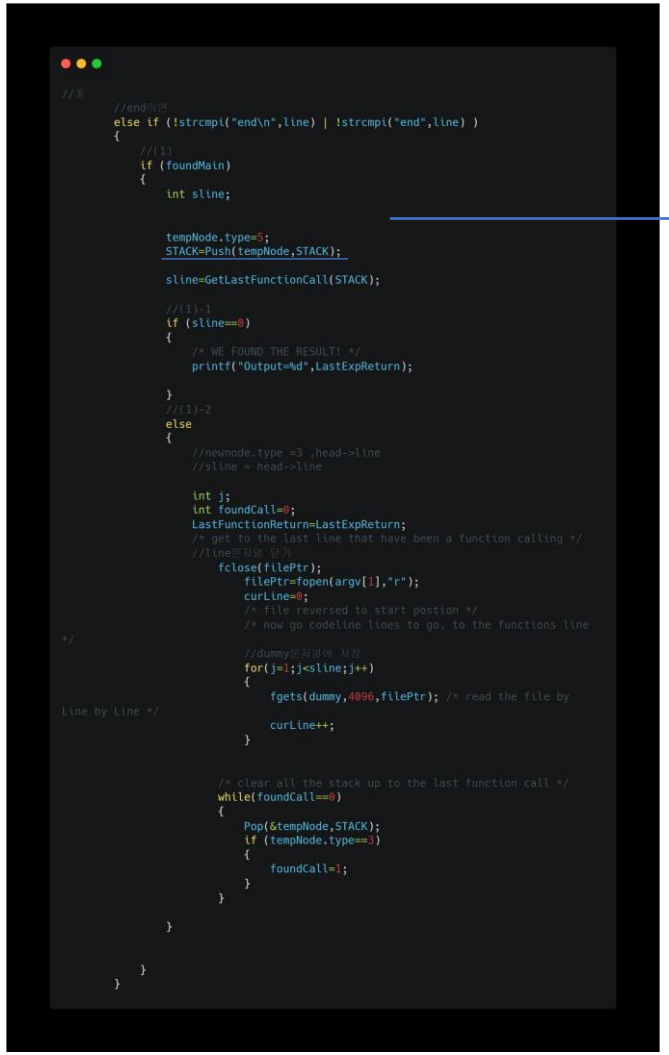
파일반복

```
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //fget을 통해 function f(int a)을 읽어 line에 저장 .
    fgets(line,4096,filePtr); /* read the file by Line by Line */
    /* scan for /t characters. get rid of them! */

    // whil문 line[k]이  null 이 아닐 시때까지 반복.
    while(line[k]!='₩0')
    {
       //  4-1) 만약 line[k]가 ₩t(탭키)일 시,
     if (line[k]=='₩t')
     {
       //line[k]는  ' '으로 바꾼다.
      line[k]=' ';
     }
      //k에 1증가
     k++;
    }
    //5. line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을  ' '로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  둘이 동일하면 1(true) 반환하고 동일하지 않으면 0(false)
    //begin과 line이 동일하다면
    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }


    //ⓑ
    //  //strcmpi :  둘이 동일하면 1(true) 반환하고 동일하지 않으면 0(false)
    //line이 end라면
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ) { ... }

    //ⓒ begin과 end가 아니면
    else{ ... }
```

input1.sql(읽을 파일)

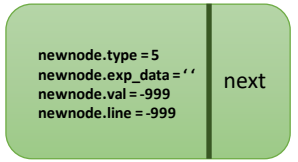
```
function f(int a)
begin
    int b = 6;
    int c = 2;
    ((b+c)/a);
end

function main()
begin
    int a = 1;
    int b = 2;
    int c = 4;
    ((6 + f(c) ) / b);
end
```

line이 end일 때



```
//8
        //end이면
        else if (!strcmpi("end\n",line) | !strcmpi("end",line) )
        {
            //(1)
            if (foundMain)
            {
                int sline;


                tempNode.type=5;
                STACK=Push(tempNode,STACK);

                sline=GetLastFunctionCall(STACK);

                //(1)-1
                if (sline==0)
                {
                    /* WE FOUND THE RESULT! */
                    printf("Output=%d",LastExpReturn);

                }
                //(1)-2
                else
                {
                    //newnode.type =3 ,head->line
                    //sline = head->line

                    int j;
                    int foundCall=0;
                    LastFunctionReturn=LastExpReturn;
                    /* get to the last line that have been a function calling */
                    //line문자열 닫기
                        fclose(filePtr);
                            filePtr=fopen(argv[1],"r");
                            curLine=0;
                            /* file reversed to start postion */
                            /* now go codeline lines to go, to the functions line
*/

                            //dummy문자열에 저장
                            for(j=1;j<sline;j++)
                            {
                                fgets(dummy,4096,filePtr); /* read the file by
Line by Line */

                                curLine++;
                            }


                        /* clear all the stack up to the last function call */
                        while(foundCall==0)
                        {
                            Pop(&tempNode,STACK);
                            if (tempNode.type==3)
                            {
                                foundCall=1;
                            }
                        }

                }


            }
        }
```

```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```

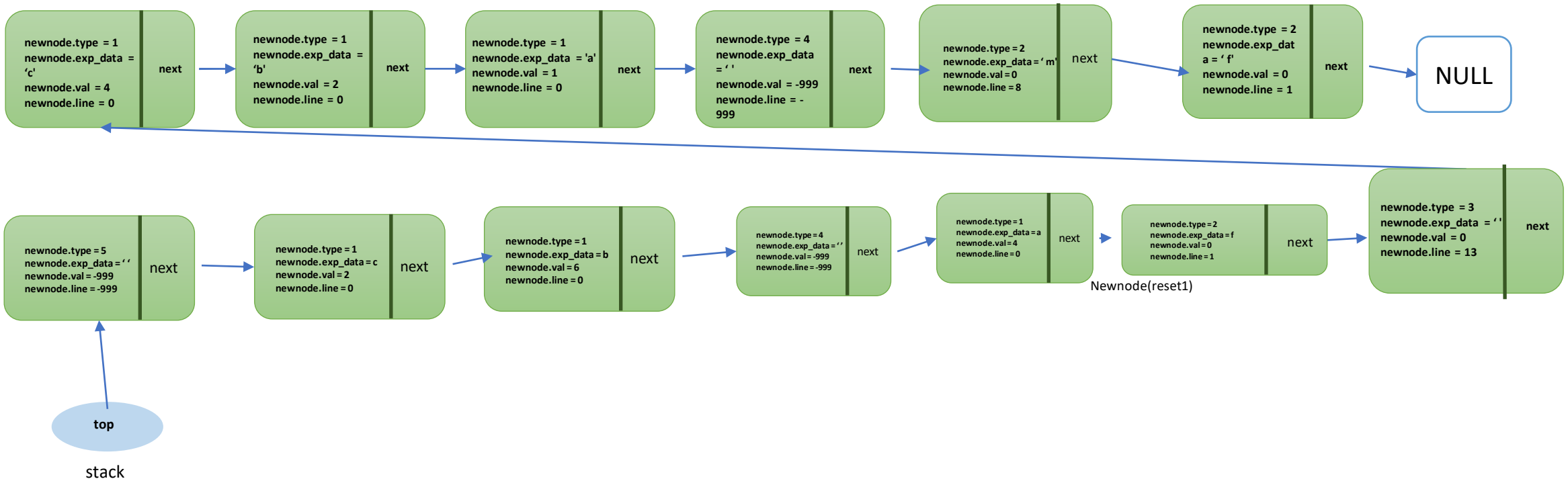


Stack * Push(Node sNode,Stack *stck)

1. Node 자료형의 newnode포인터선언;
2. 만약 newnode는 Node타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   return NULL

3.아니라면
   newnode의 type은 5의 타입이 된다.
   newnode 의 val은 -999 이된다.
   newnode 의 line 은 -999 이 된다.
   newnode 의 next 는 stck의 top이 된다.
   stck의 top은 newnode이다

반환값 stck


newnode.type = 1
newnode.exp_data = 'c'
newnode.val = 4
newnode.line = 0
next
newnode.type = 1
newnode.exp_data = 'b'
newnode.val = 2
newnode.line = 0
next
newnode.type = 1
newnode.exp_data = 'a'
newnode.val = 1
newnode.line = 0
next
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
newnode.type = 2
newnode.exp_data = 'm'
newnode.val = 0
newnode.line = 8
next
newnode.type = 2
newnode.exp_data = 'f'
newnode.val = 0
newnode.line = 1
next
NULL
newnode.type = 5
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
newnode.type = 1
newnode.exp_data = c
newnode.val = 2
newnode.line = 0
next
newnode.type = 1
newnode.exp_data = b
newnode.val = 6
newnode.line = 0
next
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
newnode.type = 1
newnode.exp_data = a
newnode.val = 4
newnode.line = 0
next
Newnode(reset1)
newnode.type = 2
newnode.exp_data = f
newnode.val = 0
newnode.line = 1
next
newnode.type = 3
newnode.exp_data = ' '
newnode.val = 0
newnode.line = 13
next
top
stack

line이 end일 때

sline=13

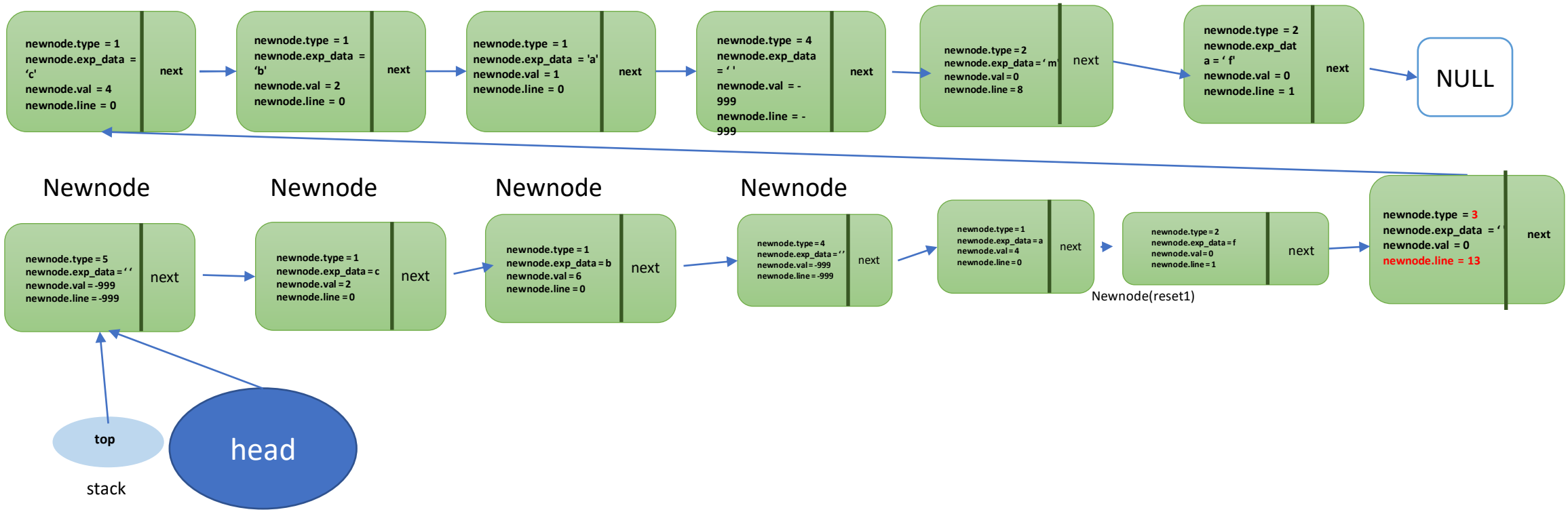
GetLastFunctionCall

```
int GetLastFunctionCall(Stack *stck)
{
Node * head;

  if (stck->top == NULL )
  {
    printf("ERROR, empty stack...");
  }
  else
  {
    head=stck->top;
    do
    {
      if ( head->type==3 )
      {
        return head->line;
      }
      else
      {
        head=head->next;
      }
    } while (head->next!=NULL);
  }

  return 0;



}
```

13

int GetLastFunctionCall(Stack *stck)

1. Node의 구조체의 자료형인 포인터 head
2. 만약 stck의 top 이 NULL이라면,
    1. ERROR, empty stack… 이라고 콘솔 출력
3. 아니라면,
    1. head는 stck의 top이다.
    2. do
        1. head의 type은 3이라면
            1. 반환값 head의 line
        2. 아니라면
            1. head는 head의 next이다.
    3. while (head의 next가 NULL이 아니라면 반복)
4. 반환값 0


newnode.type = 1
newnode.exp_data = 'c'
newnode.val = 4
newnode.line = 0
next
newnode.type = 1
newnode.exp_data = 'b'
newnode.val = 2
newnode.line = 0
next
newnode.type = 1
newnode.exp_data = 'a'
newnode.val = 1
newnode.line = 0
next
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
newnode.type = 2
newnode.exp_data = ' m'
newnode.val = 0
newnode.line = 8
next
newnode.type = 2
newnode.exp_data = ' f'
newnode.val = 0
newnode.line = 1
next
NULL
Newnode
newnode.type = 5
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
Newnode
newnode.type = 1
newnode.exp_data = c
newnode.val = 2
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = b
newnode.val = 6
newnode.line = 0
next
Newnode
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
newnode.type = 1
newnode.exp_data = a
newnode.val = 4
newnode.line = 0
next
newnode.type = 2
newnode.exp_data = f
newnode.val = 0
newnode.line = 1
next
Newnode(reset1)
newnode.type = 3
newnode.exp_data = '
newnode.val = 0
newnode.line = 13
next
top
stack
head

```
//(1)-2
else
{
    //newnode.type =3 ,head->line
    //sline = head->line

    int j;
    int foundCall=0;
    LastFunctionReturn=LastExpReturn;
    /* get to the last line that have been a function calling */
    //filePtr 닫기
        fclose(filePtr);
        /// filePtr= argv[1] 파일 읽기모드는 파일열기
            filePtr=fopen(argv[1],"r");
            curLine=0;
            /* file reversed to start postion */
            /* now go codeline lines to go, to the functions line */
            //dummy문자열에 저장
            for(j=1;j<sline;j++)
            {
                //
                fgets(dummy,4096,filePtr); /* read the file by Line by Line */
                curLine++;
            }


        /* clear all the stack up to the last function call */
        while(foundCall==0)
        {
            Pop(&tempNode,STACK);
            if (tempNode.type==3)
            {
                foundCall=1;
            }
        }

}
```

LastExpReturn = 2

LastFunctionReturn= 2

```
curLine=0;
sline=13
//dummy문자열에 저장
for(j=1;j<13;j++)
{
//
fgets(dummy,4096,filePtr); /* read the file by Line by Line */
curLine++;//12
}
```

input1.sql(읽을 파일)

```
function f(int a)
begin
   int b = 6;
   int c = 2;
   ((b+c)/a);
end

function main()
begin
   int a = 1;
   int b = 2;
   int c = 4;
   ((6 + f(c) ) / b);
end
```

```
        }

        /* clear all the stack up to the last function call */
        while(foundCall==0)
        {
            Pop(&tempNode,STACK);
            if (tempNode.type==3)
            {
                foundCall=1;
            }
        }

}
```

tempNode.type=3이나와
foundCall=1로 변경되면while문 종료

Pop

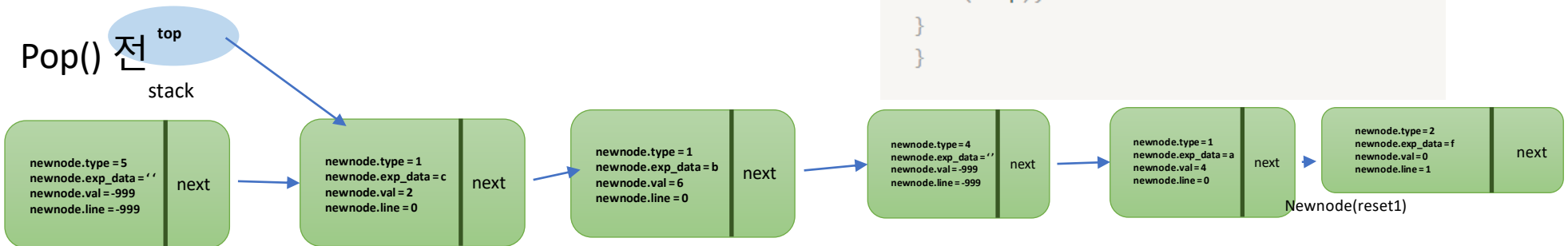
```
void Pop(Node * sNode,Stack *stck)
{
Node *temp;

if (stck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
sNode->exp_data=stck->top->exp_data;
sNode->type=stck->top->type;
sNode->line=stck->top->line;
sNode->val=stck->top->val;
temp=stck->top;
stck->top=stck->top->next;
free(temp);
}
}
```

void Pop(Node * sNode,Stack *stck)

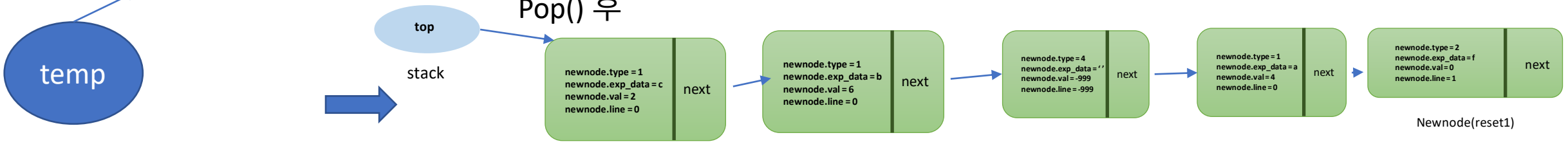
1. Node구조체 자료형인 temp 포인터 선언
2. 만약 stck의 top 이 NULL이라면,
   1. ERROR, empty stack... console출력
3. 아니라면
   1. sNode의 exp_data 는 stck의 top의 exp_data이다.
   2. sNode의 type는 stck의 top의 type 이다
   3. sNode의 line는 stck의 top의 line이다
   4. sNode의 val는 stck의 top의 val이다
   5. temp는 stck의 top이다.
   6. stck의 top은 stck의 top의 next이다.
   7. temp동적메모리 해제

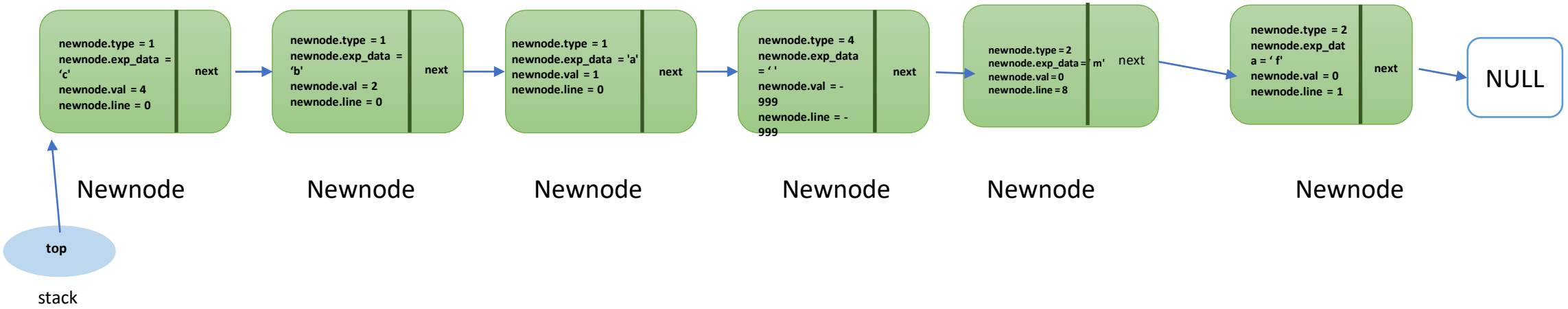
newnode.type = 5
newnode.exp_data = ‘‘
newnode.val = -999
newnode.line = -999

Pop() 전



Pop() 후




newnode.type = 1
newnode.exp_data = 'c'
newnode.val = 4
newnode.line = 0
next
newnode.type = 1
newnode.exp_data = 'b'
newnode.val = 2
newnode.line = 0
next
newnode.type = 1
newnode.exp_data = 'a'
newnode.val = 1
newnode.line = 0
next
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
newnode.type = 2
newnode.exp_data = 'm'
newnode.val = 0
newnode.line = 8
next
newnode.type = 2
newnode.exp_data = ' f'
newnode.val = 0
newnode.line = 1
next
NULL
Newnode
Newnode
Newnode
Newnode
Newnode
Newnode
top
stack

((6 + f(c)) / b);

```
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //fget을 통해 function f(int a)을 읽어 line에 저장 .
    fgets(line,4096,filePtr); /* read the file by Line by Line */
    /* scan for /t characters. get rid of them! */

    // whil문 line[k]이  null 이 아닐 시때까지 반복.
    while(line[k]!='₩0')
    {
        //  4-1) 만약 line[k]가 ₩t(탭키)일 시,
     if (line[k]=='₩t')
     {
        //line[k]는  ' '으로 바꾼다.
       line[k]=' ';
     }
      //k에 1증가
     k++;
    }
    //5. line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을  ' '로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //begin과 line이 동일하다면
    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }


    //ⓑ
    //  //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //line이 end라면
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }

    //ⓒ begin과 end가 아니면
    else{ ... }
```

input1.sql(읽을 파일)

```
function f(int a)
begin
    int b = 6;
    int c = 2;
    ((b+c)/a);
end

function main()
begin
    int a = 1;
    int b = 2;
    int c = 4;
    ((6 + f(c) ) / b);
end
```

(6 + f(c)) / b);

firstword[0]= (라면

```
//3
else if (firstword[0]=='(')
{

//foundMain == 0이면 false 0이 아니면 True
if (foundMain)
    {

    int i=0;
    int y=0; //3

    //OpStack + MathStack
    //MathStack.top은 NULL
    MathStack->top=NULL;
    /* now make the postfix calculcation */



    //(3)-2
    //lineyedek[i]이 NULL이 아닐때까지 반복
    while(lineyedek[i]!='₩x0')
    {
        //(3)-3-1
        /* evulate the function */
        //숫자라면 true
        if (isdigit(lineyedek[i])){ ... }
         /* ... */

        //(3)-3-2
        //lineyedek[i]이 ')'이라면
        else if (lineyedek[i]==')'){ ... }

        ////(3)-3-3
        //lineyedek[i]이 '+'이거나 lineyedek[i]이 '-' 이거나 lineyedek[i]이 '+' 이거나 lineyedek[i]이 '/'이라면
        else if ((lineyedek[i]=='+') | (lineyedek[i]=='-') | (lineyedek[i]=='+') | (lineyedek[i]=='/')){ ... }

        ////(3)-3-4
        //알파벳 대문자 "A-Z"는 1을 반환,알파벳 소문자 'a-z"는 2를 반환.
        //lineyedek[i] 영어라면
        else if (isalpha(lineyedek[i])>0){ ... }

        //1 증가
        i++;
    }//while문
```

```
    /* evulate the function */
    //숫자라면 true
    if (isdigit(lineyedek[i])) {
        postfix[y]=lineyedek[i];
        y++;


    }
     /*      */
```

postfix[0] = 6

(6 + f(c)) / b);

```
////(3)-3-3
//lineyedek[i]이 '+'이거나 lineyedek[i]이 '-' 이거나 lineyedek[i]이 '+' 이거나 lineyedek[i]이 '/'이라면
else if ((lineyedek[i]=='+') | (lineyedek[i]=='-') | (lineyedek[i]=='*') | (lineyedek[i]=='/'))
{
   ////(3)-3-3-1
   /*operators*/
   //00이면 false 10이면 true
   //처음일 때
   if (isStackEmpty(MathStack) != 0 ) //1일때 - stck->top==0
   {
/* if stack empty push the operator to stack */
      //+
      MathStack=PushOp(lineyedek[i],MathStack);
   }
   ////(3)-3-3-2

   //0일때 - stck -top = 00이 아닐때
   //처음이 아닐때
   else { ... }
}
```

ⓐ

ⓑ

PushOp(+, MathStack)

isStackEmpty

```
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)

1. 만약 stck의 top가 0이라면
   1. 반환값 1
2. 아니면 반환값 0

PushOp

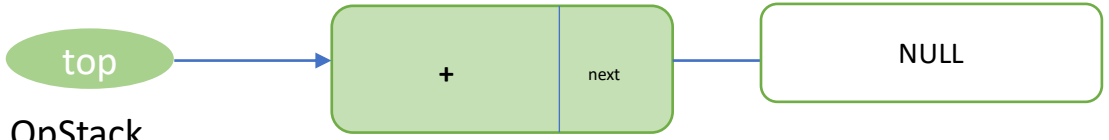
```
OpStack * PushOp(char op,OpStack *opstck)
{
opNode *newnode;
if ((newnode=(opNode*)malloc(sizeof(opNode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->op=op;
newnode->next=opstck->top;
opstck->top=newnode;
return opstck;
}
}
```

OpStack * PushOp(char op,OpStack *opstck)

1. opNode자료형의 newnode포인터선언;
2. 만약 newnode는 opNode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   반환NULL
3. 아니라면,
   1. newnode의 op는 매개변수+이다.
   2. newnode의 next는 opstck의 top이다.
   3. opstck의 top은 newnode이다.

반환값 opstck

PushOp



OpStack
MathStack

(6 + f(c)) / b);

```
//lineyedek[i] 영어라면
else if (isalpha(lineyedek[i])>0)
{
    int codeline=0;
    int dummyint=0;
    /*look if it's a variable or function call
    int retVal=0;
    retVal=GetVal(lineyedek[i],&codeline,STACK);
    //int codeline =1

    //-1


    ////(3)-3-4-1
    if ((retVal!=-1) & (retVal!=-999)){ ... }
    ////(3)-3-4-2
    //-1
    else{ ... }
}
```

GetVal (f,&codeline,STACK)

GetVal

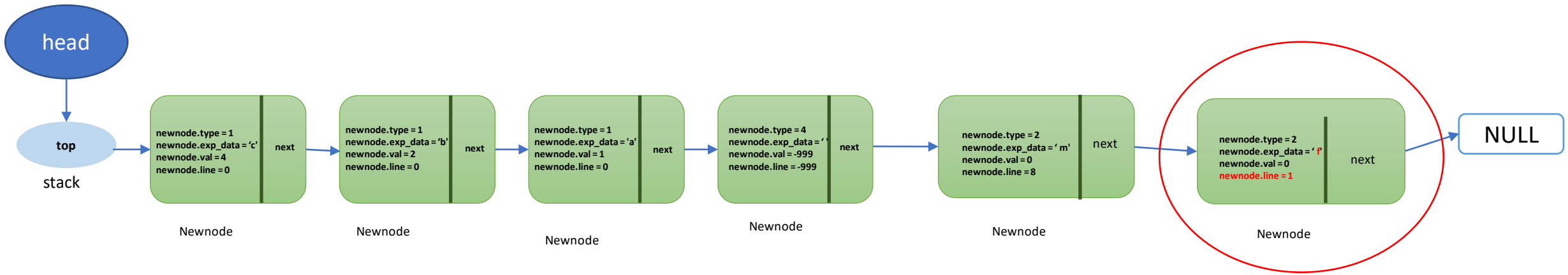
```
int GetVal(char exp_name,int * line,Stack *stck)
{

Node * head;
*line=0;
    if (stck->top == NULL )
    {
        printf("ERROR, empty stack...");
    }
    else
    {
        head=stck->top;
        do
        {
            if ( head->exp_data==exp_name )
            {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }

            }
            else
            {
                head=head->next;
            }
        } while (head->next!=NULL);
        /* check again once more */
        if (head->exp_data==exp_name)
        {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }


        }
    }

    return -999;
}
```

head->exp_data = f

*line = 1
return -1

GetVal()
자료형인 Node인 포인트 head선언

1. head를 stck의 top으로 둔다.
2. head의exp_data== exp_name
3. 와 같다면
    1. head의 type이 1이라면 head의 val이 반환값이다.
    2. head의 type이 2이라면 line은head의 line이다. -1 반환값
4. head의exp_data==exp_name
5. 와 같은게 없다면 head는 head의 next를 해준다. head의 next가 NULL이 아닐때까지 반복해준준다.



(6 + f(c)) / b);`

LastFunctionReturn= 2

```
//-1


////(3)-3-4-1
if ((retVal!=-1) & (retVal!=-999)){ ... }
////(3)-3-4-2
//-1
else
{

    ////(3)-3-4-2->1
    if (LastFunctionReturn==-999){ ... }
    ////(3)-3-4-2->2
    else

    {


    postfix[y]=LastFunctionReturn+48; /* in ascii table numeric values start from 48 */
    y++;
    i=i+3;
    LastFunctionReturn=-999;


    }
}
}

//1 증가
```

postfix[1] =50

postfix[]

6	50

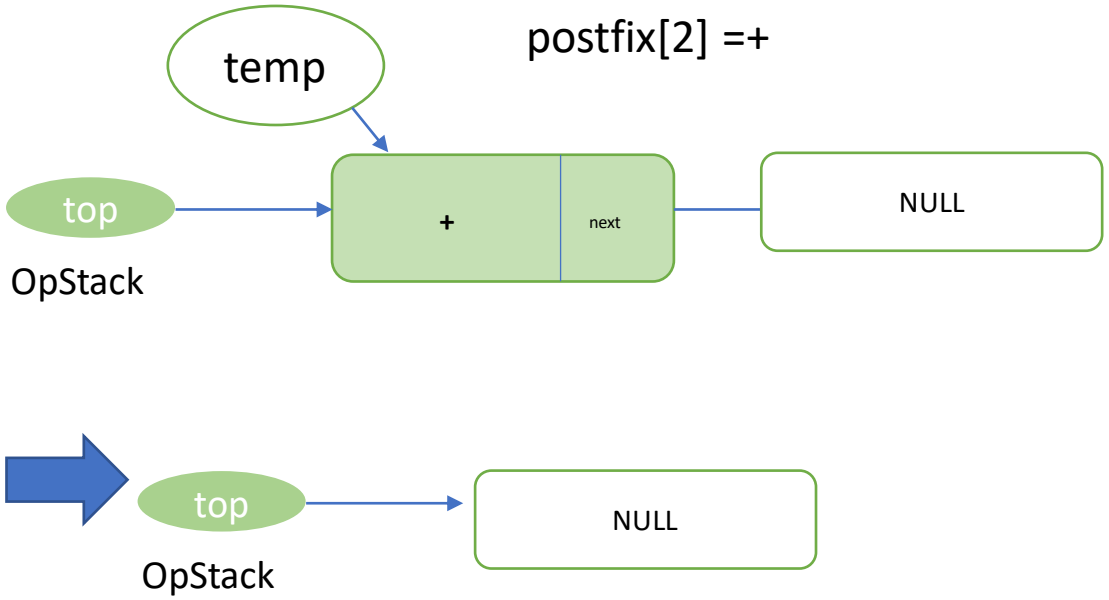
LastFunctionReturn=-999;

(6 + f(c)) / b);`

```
//lineyedek[i]이 ')'이라면
else if (lineyedek[i]==')')
{
    //(3)-3-2-1
    ////0이면 true 1이면 false
    //0일때
    if (!isStackEmpty(MathStack) != 0 )
    {
        postfix[y]=PopOp(MathStack);
        y++; // 1증가
    }
}
```

①

②



postfix[2] =+

isStackEmpty

```
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)
1. 만약 stck의 top가 0이라면
   1. 반환값 1
2. 아니면 반환값 0

PopOp

```
char PopOp(OpStack *opstck)
{
opNode *temp;
char op;
if (opstck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
op=opstck->top->op;
temp=opstck->top;
opstck->top=opstck->top->next;
free(temp);
return op;
}
return NULL;
}
```

char PopOp(OpStack *opstck)
1. opNode구조체의 자료형 temp 포인터 선언
2. 문자자료형 op선언
3. 만약 opstck의 top이 NULL일 경우
   printf("ERROR, empty stack..."); 출력

4. 아니라면,
   1. op는 + 이다
   2. temp 는 opstck의 top이다.
   3. opstck의 top은 opstck의 top의 next이다.
   temp 동적메모리 해제
   반환값 op

반환값 NULL

(6 + f(c)) / b);`

①

```
////(3)-3-3
//lineyedek[i]이 '+'이거나 lineyedek[i]이 '-' 이거나 lineyedek[i]이 '+' 이거나 lineyedek[i]이 '/'이라면
else if ((lineyedek[i]=='+') | (lineyedek[i]=='-') | (lineyedek[i]=='*') | (lineyedek[i]=='/'))
{
    ////(3)-3-3-1
    /*operators*/
    //00이면 false 10이면 true
    //처음일 때
    if (isStackEmpty(MathStack) != 0 ) // - stck->top==0
    {
/* if stack empty push the operator to stack */
      //+
      MathStack=PushOp(lineyedek[i],MathStack);
    }
    ////(3)-3-3-2
```

②

PushOp(/,MathStack)

isStackEmpty

```
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)

1. 만약 stck의 top가 0이라면
   1. 반환값 1
2. 아니면 반환값 0

PushOp

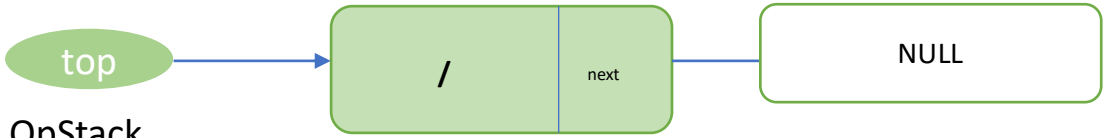
```
OpStack * PushOp(char op,OpStack *opstck)
{
opNode *newnode;
if ((newnode=(opNode*)malloc(sizeof(opNode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->op=op;
newnode->next=opstck->top;
opstck->top=newnode;
return opstck;
}
```

OpStack * PushOp(char op,OpStack *opstck)

1. opNode자료형의 newnode포인터선언;
2. 만약 newnode는 opNode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   반환NULL
3. 아니라면,
   1. newnode의 op는 매개변수op이다.
   2. newnode의 next는 opstck의 top이다.
   3. opstck의 top은 newnode이다.

반환값 opstck

PushOp



(6 + f(c)) / b);`

```
//lineyedek[i] 영어라면
else if (isalpha(lineyedek[i])>0)
{
    int codeline=0;
    int dummyint=0;
    /*look if it's a variable or function call
    int retVal=0;
    retVal=GetVal(lineyedek[i],&codeline,STACK);
    //int codeline =1

    //-1


    ////(3)-3-4-1
    if ((retVal!=-1) & (retVal!=-999)){ ... }
    ////(3)-3-4-2
    //-1
    else{ ... }
}
```

GetVal (b,&codeline,STACK)

GetVal

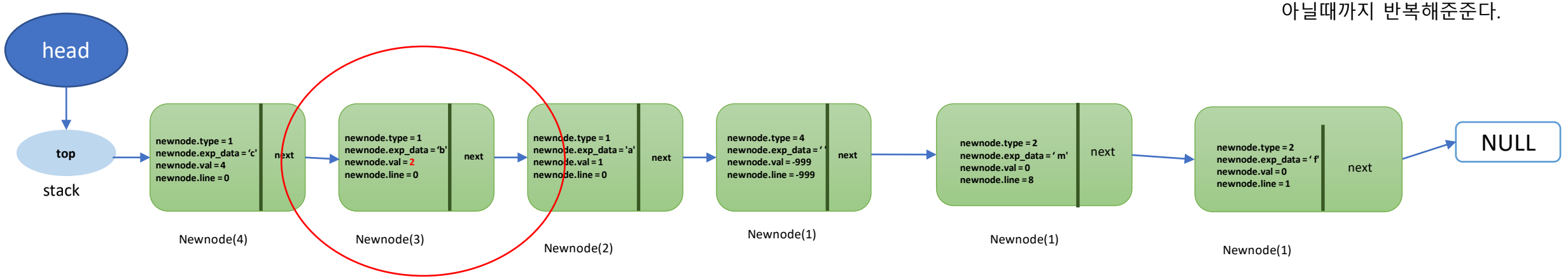
```
int GetVal(char exp_name,int * line,Stack *stck)
{
Node * head;
*line=0;
    if (stck->top == NULL )
    {
        printf("ERROR, empty stack...");
    }
    else
    {
        head=stck->top;
        do
        {
            if ( head->exp_data==exp_name )
            {
                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }

            }
            else
            {
                head=head->next;
            }
        } while (head->next!=NULL);
        /* check again once more */
        if (head->exp_data==exp_name)
        {
                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }

        }
    }
    return -999;
}
```

head->exp_data = b

return 2

GetVal()
자료형인 Node인 포인트 head선언

1. head를 stck의 top으로 둔다.
2. head의exp_data== exp_name
3. 와 같다면
   1. head의 type이 1이라면
      head의 val이 반환값이다.
   2. head의 type이 2이라면
      line은head의 line이다.
      -1 반환값
4. head의exp_data==exp_name
5. 와 같은게 없다면 head는 head의 next를 해준다. head의 next가 NULL이 아닐때까지 반복해준준다.



(6 + f(c)) / b);`

postfix[3] =50

postfix[]

6	50	+	50

```
////(3)-3-4-1
if ((retVal!=-1) & (retVal!=-999))
{
    /* if variable */
    postfix[y]=retVal+48; /* in as
    y++;

}
```

(6 + f(c)) / b);`

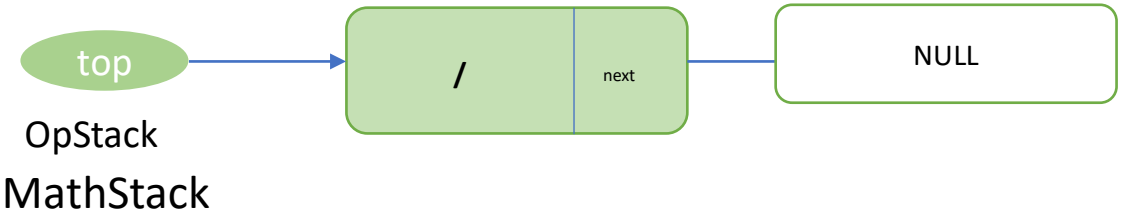
```
//(3)-3-2
//lineyedek[i]이 ')'이라면
else if (lineyedek[i]==')')
{
    //(3)-3-2-1
    ////00이면 true 10이면 false
    //0일때
    if (!isStackEmpty(MathStack) != 0 )
    {
        postfix[y]=PopOp(MathStack);
        y++; // 1증가
    }
}
```

isStackEmpty

```
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)

1. 만약 stck의 top가 0이라면
    1. 반환값 1
2. 아니면 반환값 0



OpStack

MathStack

(6 + f(c)) / b);`

```
//(3)-3-2
//lineyedek[i]이 ')'이라면
else if (lineyedek[i]==')')
{
    //(3)-3-2-1
    ////00이면 true 1이면 false
    //0일때
    if (!isStackEmpty(MathStack) != 0 )
    {
        postfix[y]=PopOp(MathStack);
        y++; // 1증가
    }
}
```

isStackEmpty

```
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)

1. 만약 stck의 top가 0이라면
   1. 반환값 1
2. 아니면 반환값 0

PopOp

```
char PopOp(OpStack *opstck)
{
opNode *temp;
char op;
if (opstck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
op=opstck->top->op;
temp=opstck->top;
opstck->top=opstck->top->next;
free(temp);
return op;
}
return NULL;
}
```

char PopOp(OpStack *opstck)

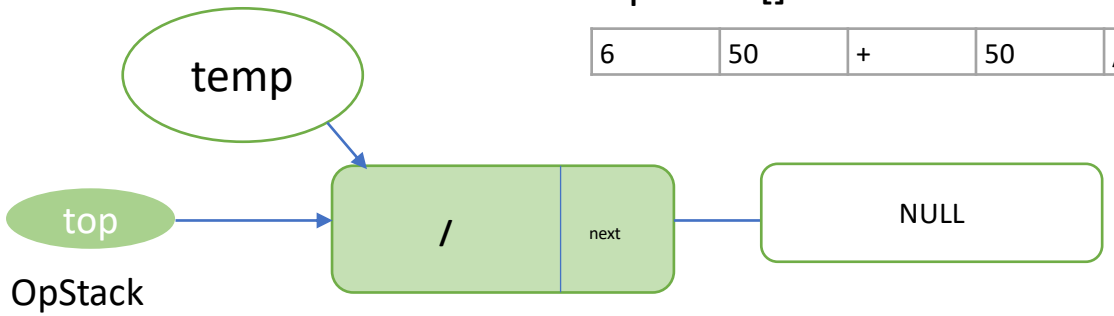
1. opNode구조체의 자료형 temp 포인터 선언
2. 문자자료형 op선언
3. 만약 opstck의 top이 NULL일 경우
   printf("ERROR, empty stack..."); 출력
4. 아니라면;
   1. op는 / 이다
   2. temp 는 opstck의 top이다.
   3. opstck의 top은 opstck의 top의 next이다.
   temp 동적메모리 해제
   반환값 op

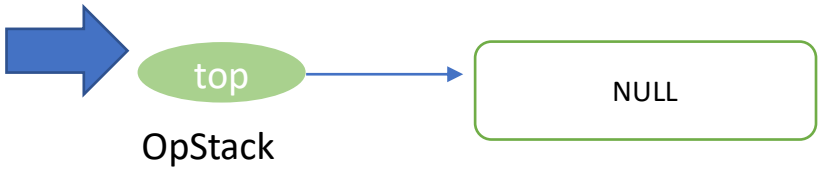
반환값 NULL

postfix[4] =/

postfix[]

6	50	+	50	/





MathStack

willBreak가 0이라면

```
//WillBreak이 0이라면
if (WillBreak==0)
{

//isStackEmpty(MathStack)이 0이라면 반복
while (isStackEmpty(MathStack)==0)
{

   /* add the poped operator to the postfix */
   //postfix[y]= PopOp(MathStack)의 반환값
   postfix[y]=PopOp(MathStack);
   //y값 1증간
   y++;
}
//postfix[y]에 '₩0'을 넣는다.
postfix[y]='₩0';
```

```c
if (WillBreak==0)
{
/* get out items left in the mathstack */
while (isStackEmpty(MathStack)==0)
{

    /* add the poped operator to the postfix */
    postfix[y]=PopOp(MathStack);
    y++;
}

postfix[y]='\0';

//MathStack=FreeAll(MathStack);

/* now calculate the postfix */
/*printf("\nCURRENT POSTFIX=%s\n",postfix);*/

i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
        val1=PopPostfix(CalcStack);

        val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;


}
```

①

②

isStackEmpty

```c
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)

1. 만약 stck의 top가 0이라면
    1. 반환값 1
2. 아니면 반환값 0

```c
char PopOp(OpStack *opstck)
{
opNode *temp;
char op;
if (opstck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
op=opstck->top->op;
temp=opstck->top;
opstck->top=opstck->top->next;
free(temp);
return op;
}
return NULL;
}
```

char PopOp(OpStack *opstck)

1. opNode구조체의 자료형 temp 포인터 선언
2. 문자자료형 op선언
3. 만약 opstck의 top이 NULL일 경우
   printf("ERROR, empty stack..."); 출력
4. 아니라면,
   1. op는 opstck의 top의 op이다
   2. temp 는 opstck의 top이다.
   3. opstck의 top은 opstck의 top의 next이다.
   temp 동적메모리 해제
   반환값 op

반환값 NULL

코드에서 실행x

```
if (WillBreak==0)
{
/* get out items left in the mathstack */
while (isStackEmpty(MathStack)==0)
{

    /* add the poped operator to the postfix */
    postfix[y]=PopOp(MathStack);
    y++;
}

postfix[y]='₩0';

//MathStack=FreeAll(MathStack);

/* now calculate the postfix */
/*printf("₩nCURRENT POSTFIX=%s₩n",postfix);*/

i=0;

CalcStack->top=NULL;
```

①

현재 MathStack
OpStack은 NULL이기에 while문 실행x

②

isStackEmpty

```
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)
1. 만약 stck의 top가 0이라면
   1. 반환값 1
2. 아니면 반환값 0

```
char PopOp(OpStack *opstck)
{
opNode *temp;
char op;
if (opstck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
op=opstck->top->op;
temp=opstck->top;
opstck->top=opstck->top->next;
free(temp);
return op;
}
return NULL;
}
```

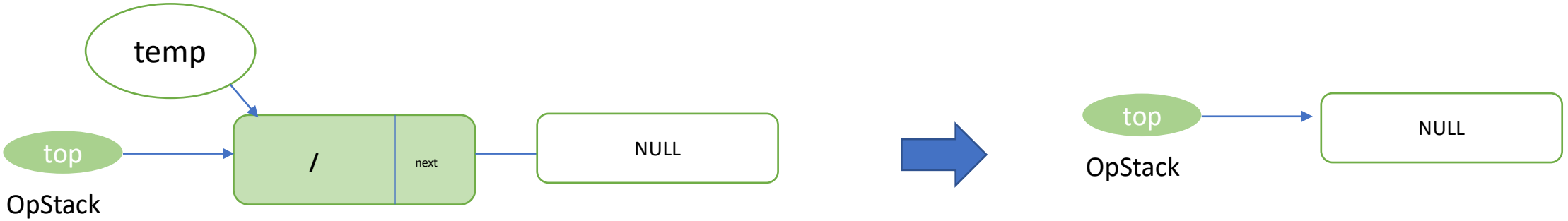
char PopOp(OpStack *opstck)
1. opNode구조체의 자료형 temp 포인터 선언
2. 문자자료형 op선언
3. 만약 opstck의 top이 NULL일 경우
   printf("ERROR, empty stack..."); 출력
4. 아니라면,
   1. op는 opstck의 top의 op이다
   2. temp 는 opstck의 top이다.
   3. opstck의 top은 opstck의 top의 next이다.
   temp 동적메모리 해제
   반환값 op

반환값 NULL

만약 Opstack stck의 top이 0이 아니라면
Opstack에 있는 값들을 PopOp함수를 통해 op값을
postfix[] 에 Opstack stck 의 top이 0 될때까지
반복하여 저장한다.

코드 실행 x

PopOp()



ⓐ `postfix[y]='\0';`

```
//MathStack=FreeAll(MathStack);

/* now calculate the postfix */
/*printf("\nCURRENT POSTFIX=%s\n",postfix);*/

i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
        val1=PopPostfix(CalcStack);

        val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

Postfix[i]이 \0일때까지 반복실행

Postfix[i]가 숫자일시,

'0'은 아스키코드 48

1증가

Postfix[]

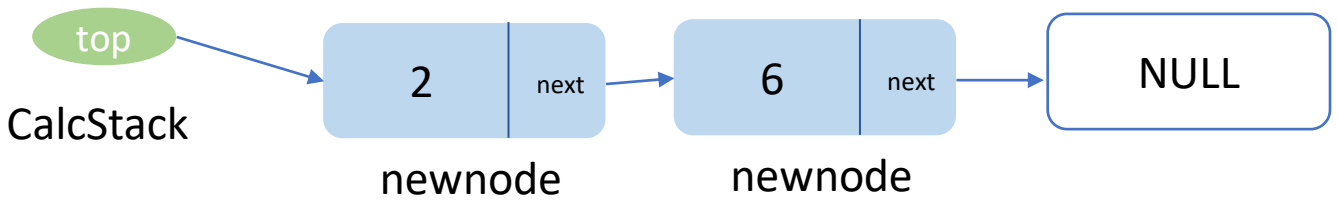
ⓐ

6	50	+	50	/	\0

```
PostfixStack * PushPostfix(int val,PostfixStack *poststck)
{
Postfixnode *newnode;
if ((newnode=(Postfixnode*)malloc(sizeof(Postfixnode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->val=val;
newnode->next=poststck->top;
poststck->top=newnode;
return poststck;
}
}
```

PostfixStack * PushPostfix(int val,PostfixStack *poststck)

1. Postfixnode구조체의 자료형인 newnode 포인터 선언
2. 만약 newnode는 Postfixnode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   NULL반환
3. 아니면라면,
   1. newnode의 val 은 매개변수 val이다
   2. newnode의 next 는 poststck의 top이다.
   3. poststck의 top 는 newnode이다.
   4. 반환값 poststck



```
i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
①       val1=PopPostfix(CalcStack);

②       val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

val1 = 2
val2=6

resultVal = 6+2

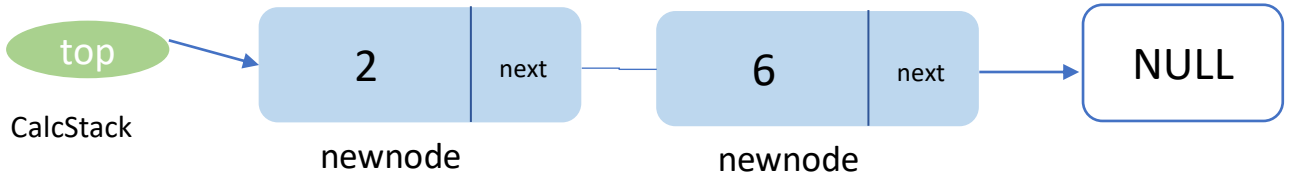
Postfix[]

6	50	+	50	/	\0

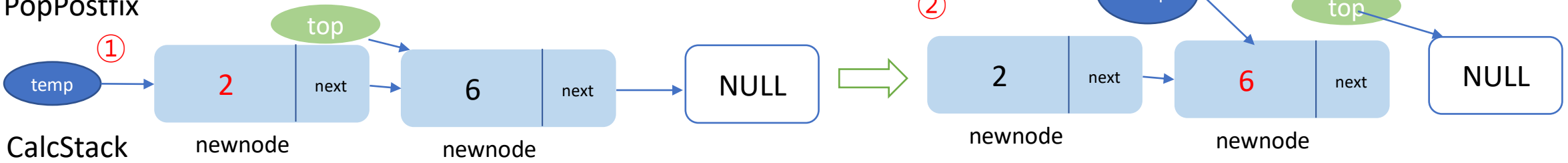
```
char PopPostfix(PostfixStack *poststck)
{
Postfixnode *temp;
int val;
if (poststck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
val=poststck->top->val;
temp=poststck->top;
poststck->top=poststck->top->next;
free(temp);
return val;
}
return NULL;
}
```

char PopPostfix(PstfixStack *poststcPostfixnode구조체의 자료k)

1. 형 temp 포인터 선언
2. 정수자료형인 val선언
3. 만약 poststck의 top이 NULL이라면
   1. ERROR, empty stack...console에 출력
4. 아니라면,
   1. val 은 poststck의 top의 val이다.
   2. tmep는 poststck의 top
   3. poststck의 top 은 poststck의 top의 next이다.
   4. temp의 동적메모리 해제
   5. 반환값 val
5. 반환값 NULL



PopPostfix



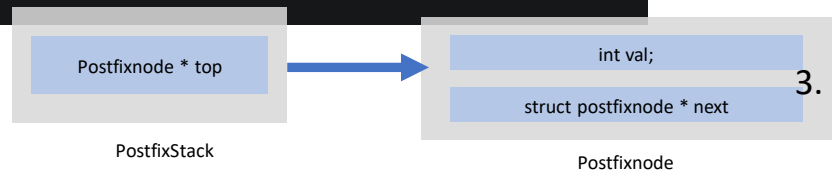
```
i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
        val1=PopPostfix(CalcStack);

        val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

pushPostfix(8,CalcStck)

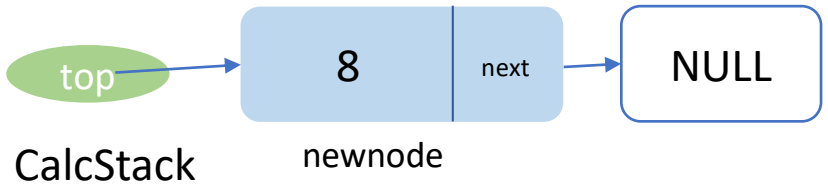
```
PostfixStack * PushPostfix(int val,PostfixStack *poststck)
{
Postfixnode *newnode;
if ((newnode=(Postfixnode*)malloc(sizeof(Postfixnode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->val=val;
newnode->next=poststck->top;
poststck->top=newnode;
return poststck;
}
}
```

PostfixStack * PushPostfix(int val,PostfixStack *poststck)

1. Postfixnode구조체의 자료형인 newnode 포인터 선언
2. 만약 newnode는 Postfixnode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   NULL반환
3. 아니면라면,
   1. newnode의 val 은 매개변수 val이다
   2. newnode의 next 는 poststck의 top이다.
   3. poststck의 top 는 newnode이다.
   4. 반환값 poststck



구조체 정리



Postfix[]

6	50	+	50	/	\0

```
postfix[y]='\0';

//MathStack=FreeAll(MathStack);

/* now calculate the postfix */
/*printf("\nCURRENT POSTFIX=%s\n",postfix);*/

i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
        val1=PopPostfix(CalcStack);

        val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

Postfix[i]이 \0일때까지 반복실행

Postfix[i]가 숫자일시,

'0'은 아스키코드 48

1증가

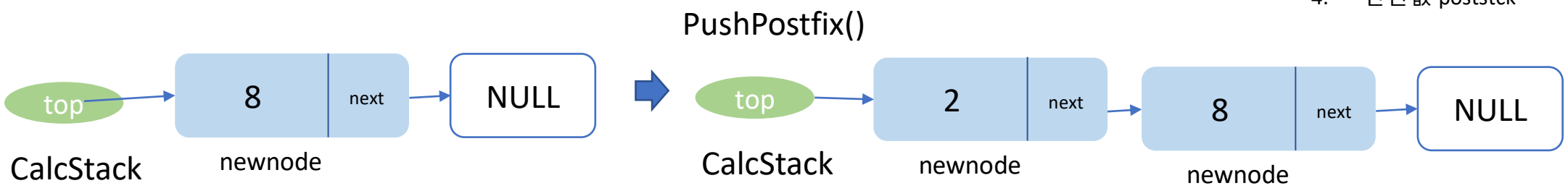
Postfix[]

6	50	+	50	/	\0

```
PostfixStack * PushPostfix(int val,PostfixStack *poststck)
{
Postfixnode *newnode;
if ((newnode=(Postfixnode*)malloc(sizeof(Postfixnode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->val=val;
newnode->next=poststck->top;
poststck->top=newnode;
return poststck;
}
}
```

PostfixStack * PushPostfix(int val,PostfixStack *poststck)

1. Postfixnode구조체의 자료형인 newnode 포인터 선언
2. 만약 newnode는 Postfixnode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   NULL반환
3. 아니면라면,
   1. newnode의 val 은 2이다
   2. newnode의 next 는 poststck의 top이다.
   3. poststck의 top 는 newnode이다.
   4. 반환값 poststck



```
i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
        val1=PopPostfix(CalcStack);

        val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

① ②

val1 = 2
val2=8

resultVal = 8/2

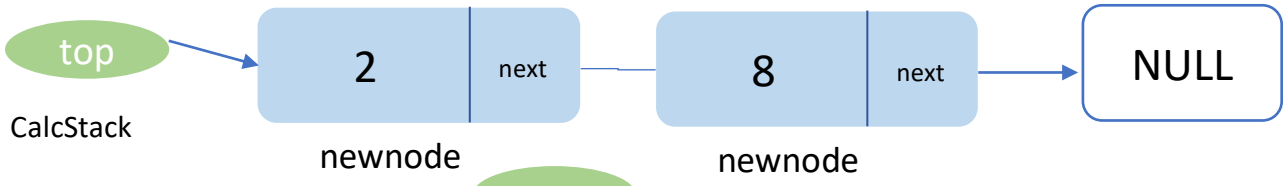
Postfix[]

6	50	'+'	52	'/'	\0

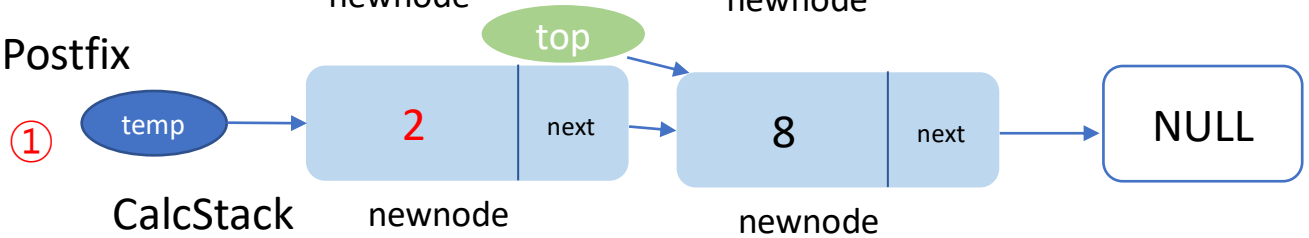
```
char PopPostfix(PostfixStack *poststck)
{
Postfixnode *temp;
int val;
if (poststck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
val=poststck->top->val;
temp=poststck->top;
poststck->top=poststck->top->next;
free(temp);
return val;
}
return NULL;
}
```

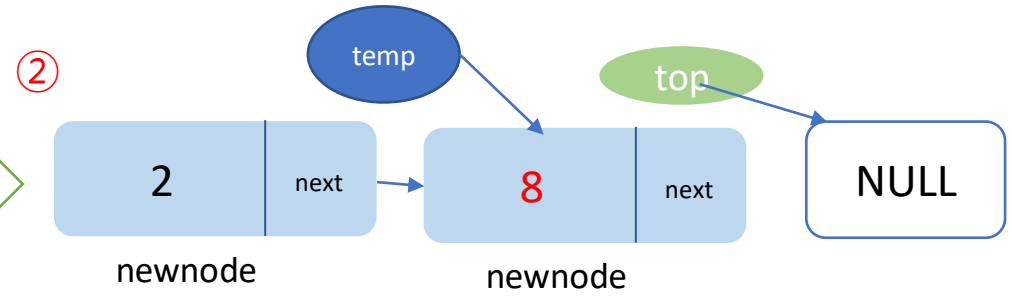
char PopPostfix(PstfixStack *poststcPostfixnode구조체의 자료k)

1. 형 temp 포인터 선언
2. 정수자료형인 val선언
3. 만약 poststck의 top이 NULL이라면
   1. ERROR, empty stack...console에 출력
4. 아니라면,
   1. val 은 poststck의 top의 val이다.
   2. tmep는 poststck의 top
   3. poststck의 top 은 poststck의 top의 next이다.
   4. temp의 동적메모리 해제
   5. 반환값 val
5. 반환값 NULL



PopPostfix





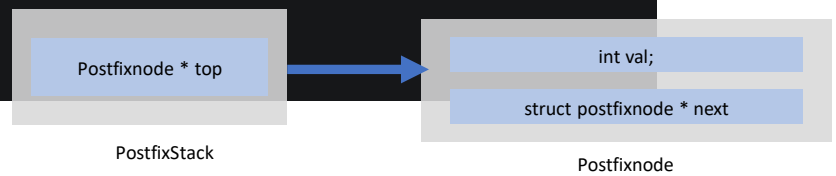
Postfix[]

6	50	'+'	52	'/'	\0

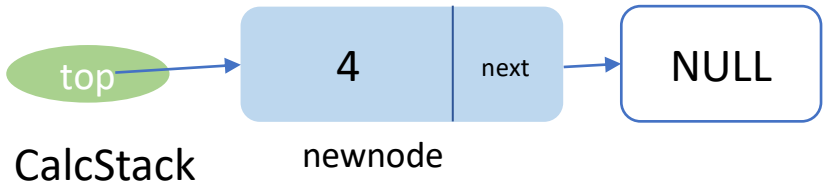
```
i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
        val1=PopPostfix(CalcStack);

        val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

pushPostfix(2,CalcStck)

```
PostfixStack * PushPostfix(int val,PostfixStack *poststck)
{
Postfixnode *newnode;
if ((newnode=(Postfixnode*)malloc(sizeof(Postfixnode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->val=val;
newnode->next=poststck->top;
poststck->top=newnode;
return poststck;
}
}
```



구조체 정리



PostfixStack * PushPostfix(int val,PostfixStack *poststck)

1. Postfixnode구조체의 자료형인 newnode 포인터 선언
2. 만약 newnode는 Postfixnode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   NULL반환
3. 아니면라면,
   1. newnode의 val 은 매개변수 val이다
   2. newnode의 next 는 poststck의 top이다.
   3. poststck의 top 는 newnode이다.
   4. 반환값 poststck

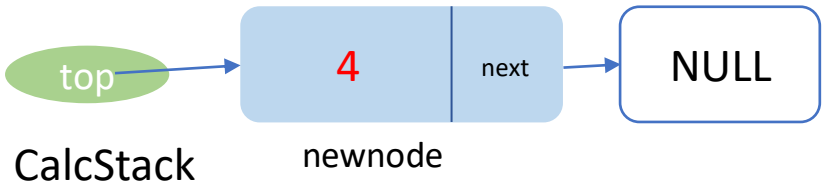
Postfix[]

6	50	'+'	52	'/'	\0

```
        postfix[y]='\0';

        //MathStack=FreeAll(MathStack);

        /* now calculate the postfix */
        /*printf("\nCURRENT POSTFIX=%s\n",postfix);*/

        i=0;

        CalcStack->top=NULL;
        while(postfix[i]!='\x0')
        {
            if (isdigit(postfix[i])) {
                /* push to stack */
                CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
            }
            else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
            {
                val1=PopPostfix(CalcStack);

                val2=PopPostfix(CalcStack);


                switch (postfix[i])
                {
                    case '+': resultVal=val2+val1;break;
                    case '-': resultVal=val2-val1;break;
                    case '/': resultVal=val2/val1;break;
                    case '*': resultVal=val2*val1;break;
                }

                CalcStack=PushPostfix(resultVal,CalcStack);
            }
            i++;
        }

        //CalcStack=FreeAll(CalcStack);
        LastExpReturn=CalcStack->top->val;
```

```
    }
    WillBreak=0;
    }
}
```

Postfix[5] = \0이기에 while반복문을 빠져나간다.



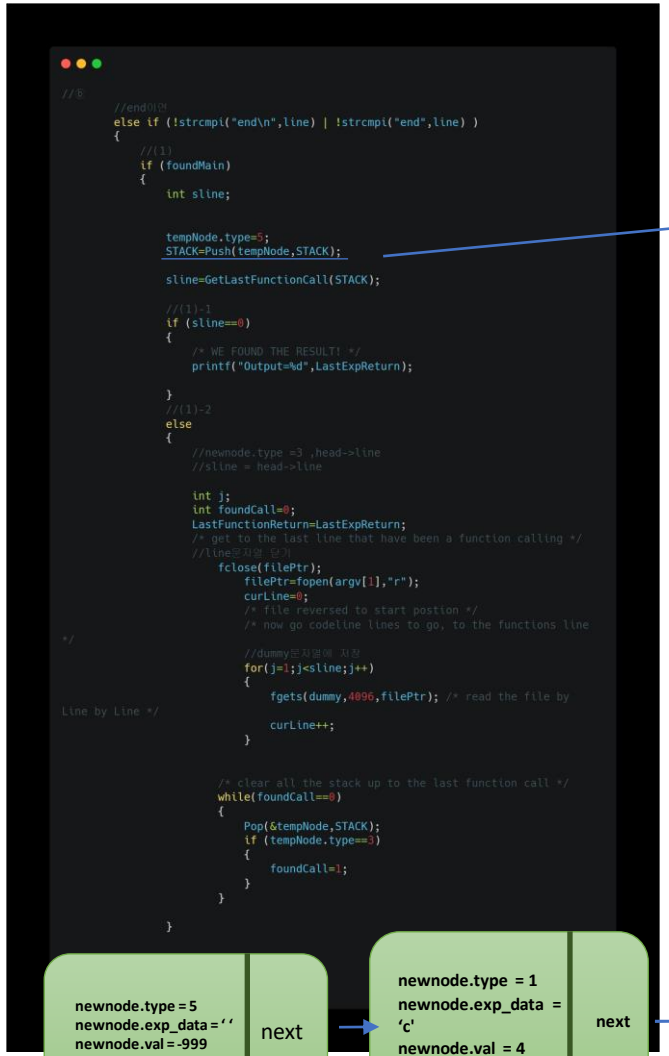
LastExpReturn= CalcStack의 top의 val이기에
LastExpReturn = 4이다.
WillBreak=0;

```
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //fget을 통해 function f(int a)을 읽어 line에 저장 .
    fgets(line,4096,filePtr); /* read the file by Line by Line */
    /* scan for /t characters. get rid of them! */

    // whil문 line[k]이  null 이 아닐 시때까지 반복.
    while(line[k]!='₩0')
    {
        //  4-1) 만약 line[k]가 ₩t(탭키)일 시,
     if (line[k]=='₩t')
     {
        //line[k]는  ' '으로 바꾼다.
       line[k]=' ';
     }
      //k에 1증가
     k++;
    }
    //5. line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을  ' '로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //begin과 line이 동일하다면
    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }


    //ⓑ
    //  //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //line이 end라면
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }

    //ⓒ begin과 end가 아니면
    else{ ... }
```

input1.sql(읽을 파일)

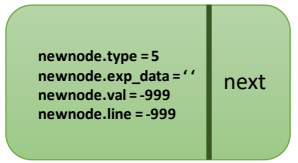
```
function f(int a)
begin
    int b = 6;
    int c = 2;
    ((b+c)/a);
end

function main()
begin
    int a = 1;
    int b = 2;
    int c = 4;
    ((6 + f(c) ) / b);
end
```

line이 end일 때



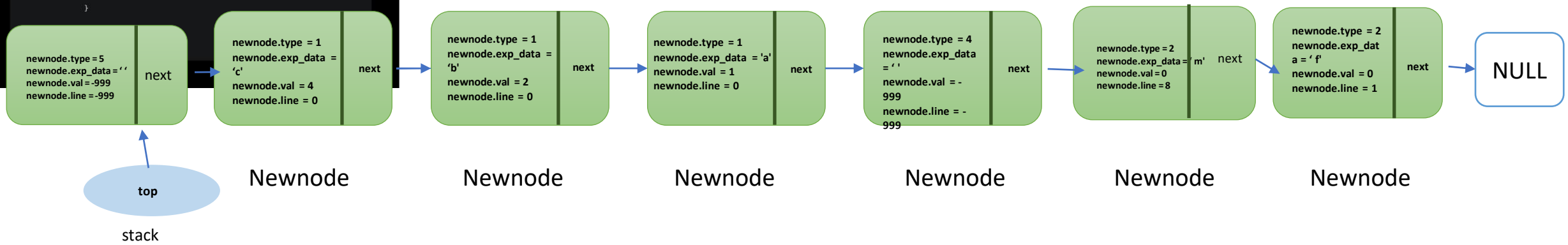
```
//8
        //end이면
        else if (!strcmpi("end\n",line) | !strcmpi("end",line) )
        {
            //(1)
            if (foundMain)
            {
                int sline;


                tempNode.type=5;
                STACK=Push(tempNode,STACK);

                sline=GetLastFunctionCall(STACK);

                //(1)-1
                if (sline==0)
                {
                    /* WE FOUND THE RESULT! */
                    printf("Output=%d",LastExpReturn);

                }
                //(1)-2
                else
                {
                    //newnode.type =3 ,head->line
                    //sline = head->line

                    int j;
                    int foundCall=0;
                    LastFunctionReturn=LastExpReturn;
                    /* get to the last line that have been a function calling */
                    //line문자열 단계
                        fclose(filePtr);
                            filePtr=fopen(argv[1],"r");
                            curLine=0;
                            /* file reversed to start postion */
                            /* now go codeline lines to go, to the functions line
*/
                            //dummy문자열에 저장
                            for(j=1;j<sline;j++)
                            {
                                fgets(dummy,4096,filePtr); /* read the file by
Line by Line */
                                curLine++;
                            }


                        /* clear all the stack up to the last function call */
                        while(foundCall==0)
                        {
                            Pop(&tempNode,STACK);
                            if (tempNode.type==3)
                            {
                                foundCall=1;
                            }
                        }

                }
```

```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```



Stack * Push(Node sNode,Stack *stck)
1. Node 자료형의 newnode포인터선언;
2. 만약 newnode는 Node타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   return NULL
3. 아니라면
   newnode의 type은 5의 타입이 된다.
   newnode 의 val은 -999 이된다.
   newnode 의 line 은 -999 이 된다.
   newnode 의 next 는 stck의 top이 된다.
   stck의 top은 newnode이다

반환값 stck



line이 end일 때

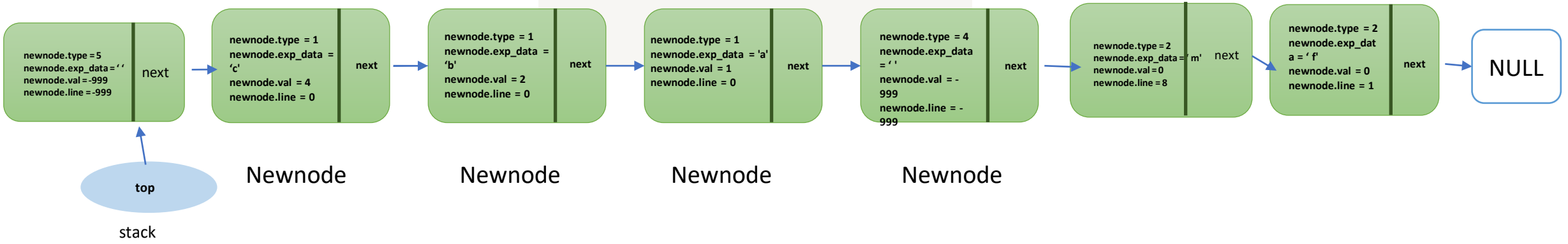
sline=0

GetLastFunctionCall

```c
int GetLastFunctionCall(Stack *stck)
{
Node * head;

  if (stck->top == NULL )
  {
    printf("ERROR, empty stack...");
  }
  else
  {
    head=stck->top;
    do
    {
      if ( head->type==3 )
      {
        return head->line;
      }
      else
      {
        head=head->next;
      }
    } while (head->next!=NULL);
  }

  return 0;
```

int GetLastFunctionCall(Stack *stck)

1. Node의 구조체의 자료형인 포인터 head
2. 만약 stck의 top 이 NULL이라면,
    1. ERROR, empty stack… 이라고 콘솔 출력
3. 아니라면,
    1. head는 stck의 top이다.
    2. do
        1. head의 type은 3이라면
            1. 반환값 head의 line
        2. 아니라면
            1. head는 head의 next이다.
    3. while (head의 next가 NULL이 아니라면 반복)
4. 반환값 0



LastExpReturn = 4

```
if (sline==0)
{
    /* WE FOUND THE RESULT! */
    // Output = LastExpReturn 콘솔 출력

    printf("Output=%d",LastExpReturn);
}
```

Output=4
콘솔에 출력

```
}
//filePtr 파일 닫기
fclose(filePtr);
//printAllStack(STACK);

STACK=FreeAll(STACK);

printf("\nPress a key to exit...");
getch();
return 0;
}
```

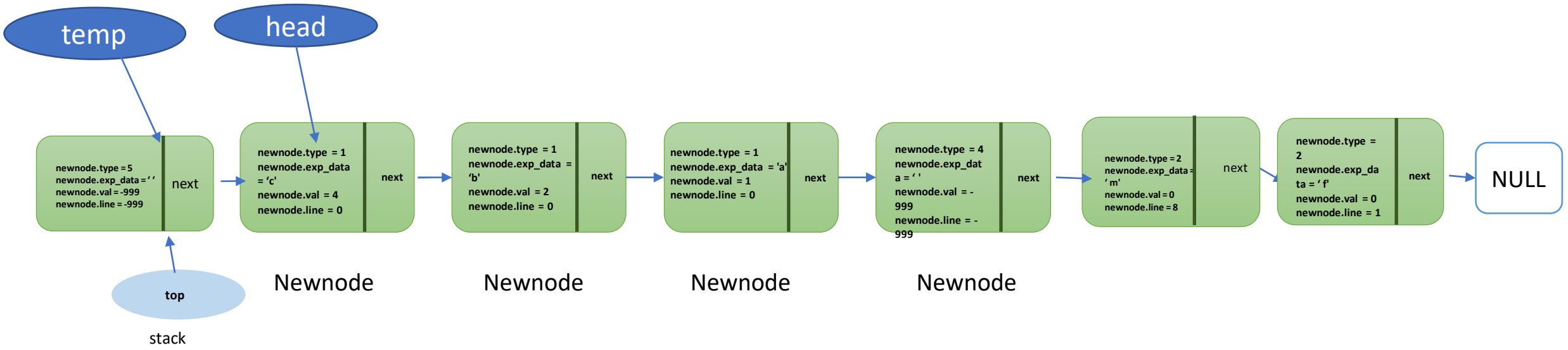
1.filePtr 파일 닫기
2.STACK= NULL
3. " Press a key to exit..."라고 콘솔에 나온다.
4. 키를 입력받는다.(input)
5. return 0

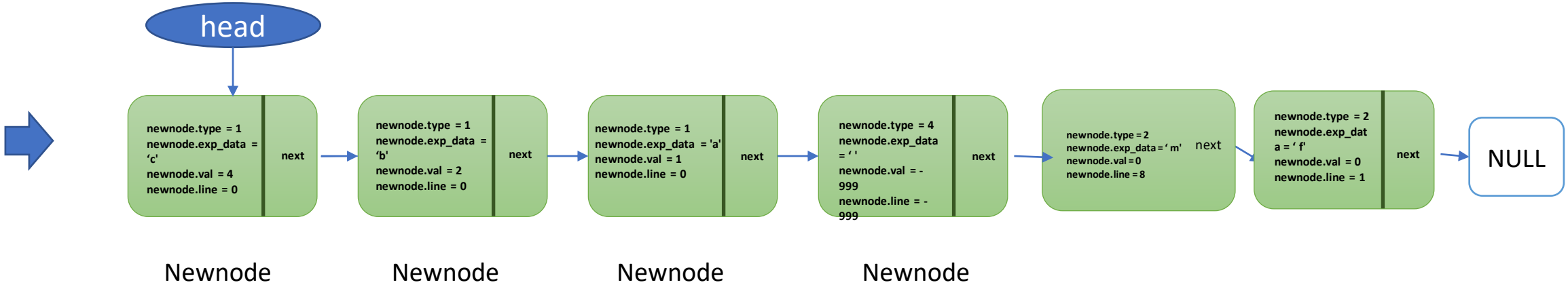
FreeAll

```
Stack * FreeAll(Stack * stck)
{
Node * temp;
Node * head;

  if (stck->top != NULL )
  {
    head=stck->top;
    do
    {

      temp=head;
      head=head->next;
      free(temp);

    } while (head->next!=NULL);
  }

return NULL;
}
```

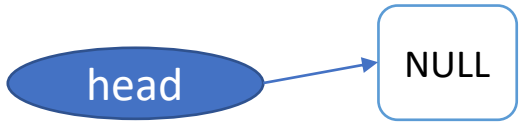
Stack * FreeAll(Stack * stck)
1. Node구조체 자료형인 포인터 temp
2. Node구조체 자료형인 포인터 head
3. 만약 stck의 top이 NULL이 아니라면,
    1. head는 stck의top이다.
    2. do
        1. temp는 head이다.
        2. head는 head의 next이다.
        3. temp 동적 메모리 할당 해제한다.
    3. while (head의 next가 NULL이 아니라면 반복)
4. 반환값 NULL

FreeAll()





head의 next가 NULL이될때까지 반복 …



Input2.sql

파일 읽기

```
//파일 스트림의 끝을 만날 때까지 반복
while (!feof(filePtr))
{

    int k=0;

    //입력 스트림에서 문자열 읽기

    fgets(line,4096, filePtr); /* read the file by Line by Line */
    /* scan for /t characters, get rid of them! */
    while(line[k]!='₩0')
    {
     if (line[k]=='₩t')
     {
        line[k]=' ';
     }

     k++;
    }

    strcpy(lineyedek,line);

    curLine++;
    tempNode.val=-999;
    tempNode.exp_data=' ';
    tempNode.line=-999;
    tempNode.type=-999;



    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }
    else{ ... }


}
```

1. 파일 스트림의 끝을 만날 때까지 반복
2. 정수 k의 값은 0
3. fget()
   fgets(파일 데이터를 저장할 변수, 읽어들일 최대 문자 수, 읽을 파일)

filePtr의파일을 최대 4095수까지 읽고 line배열에 저장.

4. whil문 line[k]이 null 이 아닐 시때까지 반복.
   4-1) 만약 line[k]가 \t(탭키)일 시, line[k]는 ‘ ‘으로 바꾼다.
   k증가

5. line을 lineyedek에 문자열 복사
6. curLine을 1증가
   tempNode.val= -999; // tempNode.val값을 -999로 대입
   tempNode.exp_data=' '; // tempNode.exp_data값을 ‘ ‘로 대입
   tempNode.line=-999;// tempNode.line 을 -999로 대입
   tempNode.type=-999;// tempNode.type 을 -999로 대입

7. 3가지로 나누어짐
   ① line이 begin일 경우
   ② line이 end일 경우
   ③ line이 begin과 end가 아닐경우

```
//파일 스트림의 끝을 만날 때까지 반복
while (!feof(filePtr))
{

    int k=0;

    //입력 스트림에서 문자열 읽기

    fgets(line,4096, filePtr); /* read the file by Line by Line */
    /* scan for /t characters. get rid of them! */
    while(line[k]!='₩0')
    {
     if (line[k]=='₩t')
     {
        line[k]=' ';
     }

     k++;
    }

    strcpy(lineyedek,line);

    curLine++;
    tempNode.val=-999;
    tempNode.exp_data=' ';
    tempNode.line=-999;
    tempNode.type=-999;



    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }
    else{ ... }


}
```

input 2.sql (읽을 파일)

```
function g(int x)
begin
 (1+2-3+x);
end

function f(int a)
begin
  int b = 1;
  int c = 2;
  ((b*c)+g(a));
end

function main()
begin
  int a = 1;
  int b = 2;
  int c = 3;
  ((2 + f(c) ) * a);
end
```

1. 파일 스트림의 끝을 만날 때까지 반복
2. 정수 k의 값은 0
3. fget을 통해 `function g(int x)`을 읽어 line에 저장 .
4. whil문 line[k]이 null 이 아닐 시때까지 반복.
   4-1) 만약 line[k]가 \t(탭키)일 시, line[k]는 ‘ ‘으로 바꾼다.
   k에 1증가
5. line을 lineyedek에 문자열 복사
6. curLine을 1증가

tempNode.val= -999; // tempNode.val값을 -999로 대입
tempNode.exp_data=' '; // tempNode.exp_data값을 ‘ ‘로 대입
tempNode.line=-999;// tempNode.line 을 -999로 대입
tempNode.type=-999;// tempNode.type 을 -999로 대입

③ line이 begin과 end가 아닐경우

```
//ⓒ begin과 end가 아니면
else
{
    //we need to tokinize
    //int
    // 공백 -> ₩0으로 바꿈
    firstword=strtok(line," ");

    //1.
    //int가 firstword이라면
    if (!strcmpi("int",firstword)) { ... }
    //2.
    else if (!strcmpi("function",firstword))
    {
        tempNode.type=2;
        firstword=strtok(NULL," ")
        tempNode.exp_data=firstword[0];
        tempNode.line=curLine;//13
        tempNode.val=0;
        STACK=Push(tempNode,STACK);

        // ...
        if ( (firstword[0]=='m') & (firstword[1]=='a') & (firstword[2]=='i') & (firstword[3]=='n') ) { ... }
        //(2)-2
        else
        {
            //(3)
            if (foundMain) { ... }

        }
```

tempNode.type=2

firstword = g(int

tempNode.exp_data = g

tempNode.line=1

tempNode. val=0

int foundMain=0임으로 false

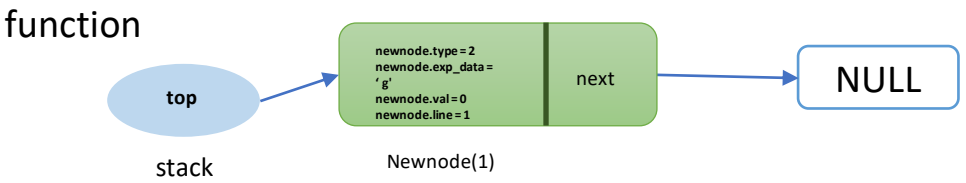
```
firstword = function
만약 function 가 firstword 이면
 temp.type = 2
firstword=strtok(NULL," ");
tempNode.exp_data=firstword[0]; //g
tempNode.line=curLine;//1
tempNode.val=0;
STACK=Push(tempNode,STACK);
```

push

```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```

Node자료형의newnode포인터선언;
만약 newnode는 Node타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우 ERROR, Couldn't allocate memory... 출력
NULL리턴

newnode.type = 2
newnode.val = 0
newnode.exp_data = g
newnode.line = 1
newnode 의 next 는 stck의 top이 된다.
stck의 top은 newnode이다
반환값 stck



```
//파일 스트림의 끝을 만날 때까지 반복
while (!feof(filePtr))
{

    int k=0;

    //입력 스트림에서 문자열 읽기

    fgets(line,4096, filePtr); /* read the file by Line by Line */
    /* scan for /t characters. get rid of them! */
    while(line[k]!='₩0')
    {
     if (line[k]=='₩t')
     {
        line[k]=' ';
     }

     k++;
    }

    strcpy(lineyedek,line);

    curLine++;
    tempNode.val=-999;
    tempNode.exp_data=' ';
    tempNode.line=-999;
    tempNode.type=-999;



    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }
    else{ ... }


}
```

curLine =2

input 2.sql (읽을 파일)

```
function g(int x)
begin
 (1+2-3+x);
end

function f(int a)
begin
  int b = 1;
  int c = 2;
  ((b*c)+g(a));
end

function main()
begin
  int a = 1;
  int b = 2;
  int c = 3;
  ((2 + f(c) ) * a);
end
```

1. 파일 스트림의 끝을 만날 때까지 반복
2. 정수 k의 값은 0
3. fget을 통해 begin 을 읽어 line에 저장 .
4. whil문 line[k]이 null 이 아닐 시때까지 반복.
   4-1) 만약 line[k]가 \t(탭키)일 시, line[k]는 ‘ ‘으로 바꾼다.
   k증가
5. line을 lineyedek에 문자열 복사
6. curLine을 1증가

tempNode.val= -999; // tempNode.val값을 -999로 대입
tempNode.exp_data=' '; // tempNode.exp_data값을 ‘ ‘로 대입
tempNode.line=-999;// tempNode.line 을 -999로 대입
tempNode.type=-999;// tempNode.type 을 -999로 대입

① line이 begin일 경우

```
// @하나
        //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
        //begind이면
        if (!strcmpi("begin\n",line) | !strcmpi("begin",line))
        {
            //foundMain == 0이면 false 0이 아니면 True
            if (foundMain)
            {
                tempNode.type=4;
                STACK=Push(tempNode,STACK);
            }
        }
```

int foundMain=0임으로 false

```
//파일 스트림의 끝을 만날 때까지 반복
while (!feof(filePtr))
{

    int k=0;

    //입력 스트림에서 문자열 읽기

    fgets(line,4096, filePtr); /* read the file by Line by Line */
    /* scan for /t characters. get rid of them! */
    while(line[k]!='₩0')
    {
    if (line[k]=='₩t')
    {
       line[k]=' ';
    }

    k++;
    }

    strcpy(lineyedek,line);

    curLine++;
    tempNode.val=-999;
    tempNode.exp_data=' ';
    tempNode.line=-999;
    tempNode.type=-999;



    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }
    else{ ... }


}
```

curLine=3

input 2.sql (읽을 파일)

```
function g(int x)
begin
 (1+2-3+x);
end

function f(int a)
begin
  int b = 1;
  int c = 2;
  ((b*c)+g(a));
end

function main()
begin
  int a = 1;
  int b = 2;
  int c = 3;
  ((2 + f(c) ) * a);
end
```

1. 파일 스트림의 끝을 만날 때까지 반복
2. 정수 k의 값은 0
3. fget을 통해 (1+2-3+x); 을 읽어 line에 저장 .
4. whil문 line[k]이 null 이 아닐 시때까지 반복.
   4-1) 만약 line[k]가 \t(탭키)일 시, line[k]는 ' '으로 바꾼다.
   k증가
5. line을 lineyedek에 문자열 복사
6. curLine을 1증가

tempNode.val= -999; // tempNode.val값을 -999로 대입
tempNode.exp_data=' '; // tempNode.exp_data값을 ' '로 대입
tempNode.line=-999;// tempNode.line 을 -999로 대입
tempNode.type=-999;// tempNode.type 을 -999로 대입

③ line이 begin과 end가 아닐경우

```
//ⓒ begin과 end가 아니면
else
{
    //we need to tokinize
    //int
    // 공백 -> \0으로 바꿈
    firstword=strtok(line," ");

    //1.
    //int가 firstword이라면
    if (!strcmpi("int",firstword))
    {
        //(1)-1
        //foundMain ==0
        //foundMain == 0이면 false 0이 아니면 True
        if (foundMain) { ... }
    }
    //2.
    else if (!strcmpi("function",firstword)) { ... }

    //3
    else if (firstword[0]=='(')
    {
    //(3)-1
    if (foundMain) { ... }
    }
}
```

firstword =(

int foundMain=0임으로 false

```
//파일의 끝에 도달했는지 여부를 확인
//파일의 끝에 도달하지 못했을 경우에 0 반환
//파일 스트림의 끝을 만날 때까지 반복
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //fget을 통해 function f(int a)을 읽어 line에 저장 ,
    fgets(line,4096,filePtr); /* read the file by Line by Line */
    /* scan for /t characters, get rid of them! */
    // while문 line[k]이  null 이 아닐 시때까지 반복.

    while(line[k]!='₩0')
    {
       //  4-1) 만약 line[k]가 ₩t(탭키)일 시,
     if (line[k]=='₩t')
     {
        //line[k]는  ' '으로 바꾼다.
       line[k]=' ';
     }
      //k에 1증가
     k++;
    }
    //5, line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을  ' '로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //begin과 line이 동일하다면
    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }


    //ⓑ
    //  //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //line이 end라면
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }

    //ⓒ begin과 end가 아니면
    else{ ... }


}
```

input 2.sql (읽을 파일)

```
function g(int x)
begin
 (1+2-3+x);
end

function f(int a)
begin
  int b = 1;
  int c = 2;
  ((b*c)+g(a));
end

function main()
begin
  int a = 1;
  int b = 2;
  int c = 3;
  ((2 + f(c) ) * a);
end
```

```
//ⓑ
//  //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
//line이 end라면
else if (!strcmpi("end₩n",line) | !strcmpi("end",line) )
{
    //(1)

    //foundMain == 0이면  false 0이 아니면 True
    if (foundMain){ ... }
}
```

int foundMain=0임으로 false

```
//파일 스트림의 끝을 만날 때까지 반복
while (!feof(filePtr))
{

    int k=0;

    //입력 스트림에서 문자열 읽기

    fgets(line,4096, filePtr); /* read the file by Line by Line */
    /* scan for /t characters. get rid of them! */
    while(line[k]!='₩0')
    {
     if (line[k]=='₩t')
     {
        line[k]=' ';
     }

     k++;
    }

    strcpy(lineyedek,line);

    curLine++;
    tempNode.val=-999;
    tempNode.exp_data=' ';
    tempNode.line=-999;
    tempNode.type=-999;



    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }
    else{ ... }


}
```

input3.sql(읽을 파일)

```
function g(int x)
begin
  (1+2-3+x);
end

function f(int a)
begin
   int b = 1;
   int c = 2;
   if ( b < (b+c))
   ((b*c)+g(a));
end

function main()
begin
   int a = 1;
   int b = 2;
   int c = 3;
   ((2 + f(c) ) * a);
end
```

1. 파일 스트림의 끝을 만날 때까지 반복
2. 정수 k의 값은 0
3. fget을 통해 function f(int a) 을 읽어 line에 저장 .
4. whil문 line[k]이 null 이 아닐 시때까지 반복.
   4-1) 만약 line[k]가 \t(탭키)일 시, line[k]는 ' '으로 바꾼다.
   k에 1증가
5. line을 lineyedek에 문자열 복사
6. curLine을 1증가

tempNode.val= -999; // tempNode.val값을 -999로 대입
tempNode.exp_data=' '; // tempNode.exp_data값을 ' '로 대입
tempNode.line=-999;// tempNode.line 을 -999로 대입
tempNode.type=-999;// tempNode.type 을 -999로 대입

③ line이 begin과 end가 아닐경우

```
//© begin과 end가 아니면
else
{
    //we need to tokinize
    //int
    // 공백 -> \0으로 바꿈
    firstword=strtok(line," ");

    //1.
    //int가 firstword이라면
    if (!strcmpi("int",firstword)) { ... }
    //2.
    else if (!strcmpi("function",firstword))
    {
        tempNode.type=2;
        firstword=strtok(NULL," "); //f(int

        tempNode.exp_data=firstword[0];
        tempNode.line=curLine;//13
        tempNode.val=0;
        STACK=Push(tempNode,STACK);

        // ...
        if ( (firstword[0]=='m') & (firstword[1]=='a') & (firstword[2]=='i') & (firstword[3]=='n') ) { ... }
        //(2)-2
        else
        {
            //(3)
            if (foundMain) { ... }

        }
```

tempNode.type=2

firstword = f(int

tempNode.exp_data = f

tempNode.line=6

tempNode. val=0

int foundMain=0임으로 false

```
firstword = function
만약 function 가 firstword 이면
 temp.type = 2
firstword=strtok(NULL," ");
tempNode.exp_data=firstword[0]; //f
tempNode.line=curLine;//6
tempNode.val=0;
STACK=Push(tempNode,STACK);
```

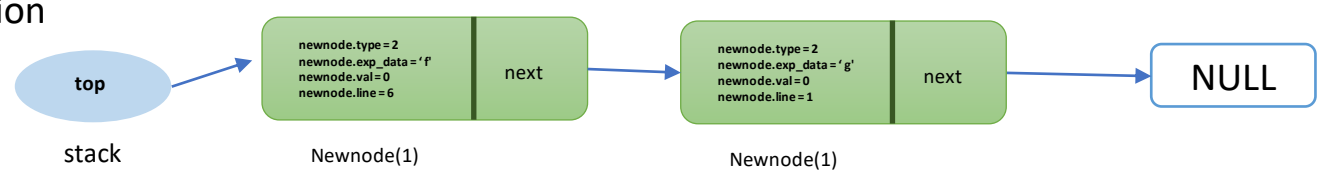
push

```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```

Node자료형의newnode포인터선언;
만약 newnode는 Node타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우 ERROR, Couldn't allocate memory... 출력
NULL리턴

newnode.type = 2
newnode.val = 0
newnode.exp_data = f
newnode.line = 6
newnode 의 next 는 stck의 top이 된다.
stck의 top은 newnode이다
반환값 stck

function



```
//파일 스트림의 끝을 만날 때까지 반복
while (!feof(filePtr))
{

    int k=0;

    //입력 스트림에서 문자열 읽기

    fgets(line,4096, filePtr); /* read the file by Line by Line */
    /* scan for /t characters. get rid of them! */
    while(line[k]!='₩0')
    {
     if (line[k]=='₩t')
     {
        line[k]=' ';
     }

     k++;
    }

    strcpy(lineyedek,line);

    curLine++;
    tempNode.val=-999;
    tempNode.exp_data=' ';
    tempNode.line=-999;
    tempNode.type=-999;



    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }
    else{ ... }


}
```

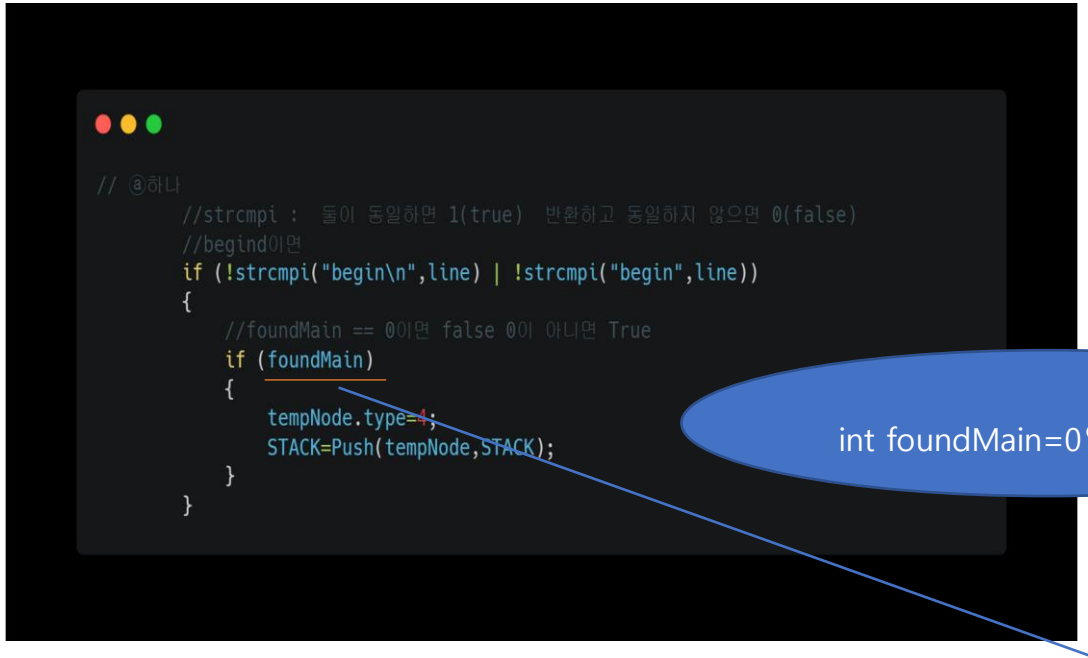
input 2.sql (읽을 파일)

```
function g(int x)
begin
 (1+2-3+x);
end

function f(int a)
begin
  int b = 1;
  int c = 2;
  ((b*c)+g(a));
end

function main()
begin
  int a = 1;
  int b = 2;
  int c = 3;
  ((2 + f(c) ) * a);
end
```

1. 파일 스트림의 끝을 만날 때까지 반복
2. 정수 k의 값은 0
3. fget을 통해 begin 을 읽어 line에 저장 .
4. whil문 line[k]이 null 이 아닐 시때까지 반복.
   4-1) 만약 line[k]가 \t(탭키)일 시, line[k]는 ‘ ‘으로 바꾼다.
   k증가
5. line을 lineyedek에 문자열 복사
6. curLine을 1증가

tempNode.val= -999; // tempNode.val값을 -999로 대입
tempNode.exp_data=' '; // tempNode.exp_data값을 ‘ ‘로 대입
tempNode.line=-999;// tempNode.line 을 -999로 대입
tempNode.type=-999;// tempNode.type 을 -999로 대입

① line이 begin일 경우



```
// @하나
        //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
        //begind이면
        if (!strcmpi("begin\n",line) | !strcmpi("begin",line))
        {
            //foundMain == 0이면 false 0이 아니면 True
            if (foundMain)
            {
                tempNode.type=4;
                STACK=Push(tempNode,STACK);
            }
        }
```

int foundMain=0임으로 false

main을 찾을때까지
False이다.

③ line이 begin과 end가 아닐경우

input 2.sql (읽을 파일)

```
function g(int x)
begin
 (1+2-3+x);
end

function f(int a)
begin
  int b = 1;
  int c = 2;
  ((b*c)+g(a));
end

function main()
begin
  int a = 1;
  int b = 2;
  int c = 3;
  ((2 + f(c) ) * a);
end
```

```
else
{
    //we need to tokinize
    //int
    // 공백 -> 0으로 바꿈
    firstword=strtok(line," ");

    //1.
    //int가 firstword이라면
    if (!strcmpi("int",firstword))
    {
        //(1)-1
        //foundMain ==0
        //foundMain == 0이면 false 0이 아니면 True
        if (foundMain){ ... }
    }
    //2.
```

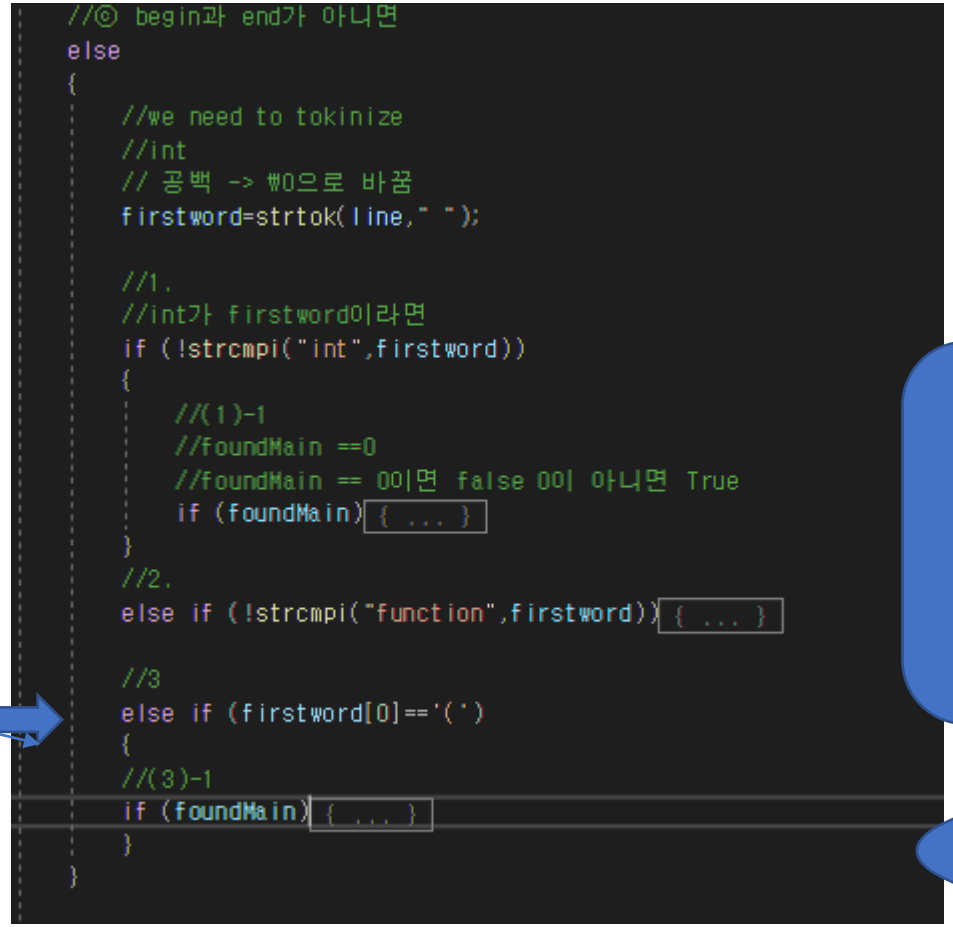
firstword =int

int foundMain=0임으로 false

③ line이 begin과 end가 아닐경우

input 2.sql (읽을 파일)

```
function g(int x)
begin
 (1+2-3+x);
end

function f(int a)
begin
  int b = 1;
  int c = 2;
  ((b*c)+g(a));
end

function main()
begin
  int a = 1;
  int b = 2;
  int c = 3;
  ((2 + f(c) ) * a);
end
```



firstword =(

int foundMain=0임으로 false

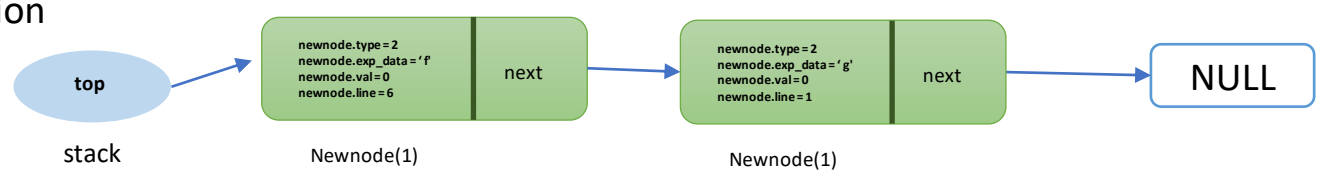
function main()

```
//파일의 끝에 도달했는지 여부를 확인
//파일의 끝에 도달하지 못했을 경우에 0 반환
//파일 스트림의 끝을 만날 때까지 반복
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //fget을 통해 function f(int a)을 읽어 line에 저장 .
    fgets(line,4096,filePtr); /* read the file by Line by Line */
    /* scan for /t characters, get rid of them! */
    // while문 line[k]이  null 이 아닐 시때까지 반복.

    while(line[k]!='₩0')
    {
        //  4-1) 만약 line[k]가 ₩t(탭키)일 시,
     if (line[k]=='₩t')
     {
         //line[k]는  ' '으로 바꾼다.
       line[k]=' ';
     }
      //k에 1증가
     k++;
    }
    //5. line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을  ' '로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //begin과 line이 동일하다면
    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)) { ... }


    //ⓑ
    //  //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //line이 end라면
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ) { ... }

    //ⓒ begin과 end가 아니면
    else { ... }


}
```

input 2.sql (읽을 파일)

```
function g(int x)
begin
  (1+2-3+x);
end

function f(int a)
begin
   int b = 1;
   int c = 2;
   ((b*c)+g(a));
end

function main()
begin
   int a = 1;
   int b = 2;
   int c = 3;
   ((2 + f(c) ) * a);
end
```

function



```
else if (!strcmpi("function",firstword))
{
  //tempNode.type는 2이다
  tempNode.type=2;

  //  //자른 문자 다음부터 구분자 찾기
  firstword=strtok(NULL," ");

  //tempNode.exp_data는 firstword[0]이다
  tempNode.exp_data=firstword[0];

  //tempNode.line 는 curLine이다
  tempNode.line=curLine;
  //tempNode.val은 0이다.
  tempNode.val=0;
  //STACK은 Push(tempNode,STACK); 에서의 반환값이다.
  STACK=Push(tempNode,STACK);

  //main
  //(2)-1
  //만약 firstword배열의 0번째가 'm'이며 firstword배열의 1번째가 'a'이며 firstword배열의 2번째가 'i'이며 firstword배열의 3번째가 'n'이라면
  if ( (firstword[0]=='m') & (firstword[1]=='a') & (firstword[2]=='i') & (firstword[3]=='n') )
  {
    /*printf("Found function main() in line %d. Starting to running the script...₩n",curLine);*/
    //foundMain 은 1이다.
    foundMain=1;

  }
```

tempNode.type=2

firstword = main()

tempNode.exp_data = m

tempNode.line=13

tempNode. val=0

foundMain =1

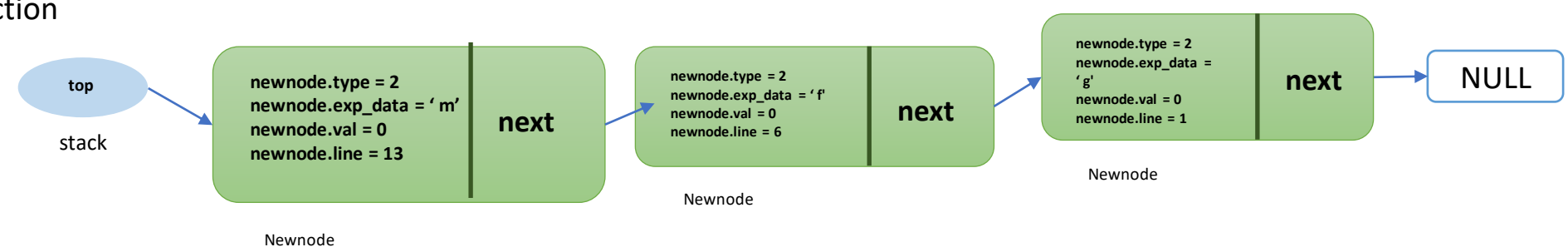
Push

```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```

Node자료형의newnode포인터선언;
만약 newnode는 Node타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우 ERROR, Couldn't allocate memory... 출력
NULL리턴

newnode.type = 2
newnode.val = 0
newnode.exp_data = ' m'
newnode.line = 8
newnode 의 next 는 stck의 top이 된다.
stck의 top은 newnode이다
반환값 stck

function



```
//파일의 끝에 도달했는지 여부를 확인
//파일의 끝에 도달하지 못했을 경우에 0 반환
//파일 스트림의 끝을 만날 때까지 반복
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //fget을 통해 function f(int a)을 읽어 line에 저장 .
    fgets(line,4096,filePtr); /* read the file by Line by Line */
    /* scan for /t characters, get rid of them! */
    // while문 line[k]이  null 이 아닐 시때까지 반복.

    while(line[k]!='₩0')
    {
       //  4-1) 만약 line[k]가 ₩t(탭키)일 시,
     if (line[k]=='₩t')
     {
        //line[k]는  ' '으로 바꾼다.
       line[k]=' ';
     }
      //k에 1증가
     k++;
    }
    //5, line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을  ' '로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //begin과 line이 동일하다면
    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }


    //ⓑ
    //  //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //line이 end라면
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }

    //ⓒ begin과 end가 아니면
    else{ ... }


}
```

input 2.sql (읽을 파일)

```
function g(int x)
begin
  (1+2-3+x);
end

function f(int a)
begin
   int b = 1;
   int c = 2;
   ((b*c)+g(a));
end

function main()
begin
   int a = 1;
   int b = 2;
   int c = 3;
   ((2 + f(c) ) * a);
end
```

begin

```
// ⓐ하나
//strcmpi :  둘이 동일하면 1(true) 반환하고 동일하지 않으면 0(false)
//begin과 line이 동일하다면
if (!strcmpi("beginMn",line) | !strcmpi("begin",line))
{
    //foundMain == 0이면 false 0이 아니면 True
    if (foundMain)
    {
        //tempNode.type 는 4이다.
        tempNode.type=4;
        ////STACK은 Push(tempNode,STACK); 에서의 반환값이다.
        STACK=Push(tempNode,STACK);
    }
}
```

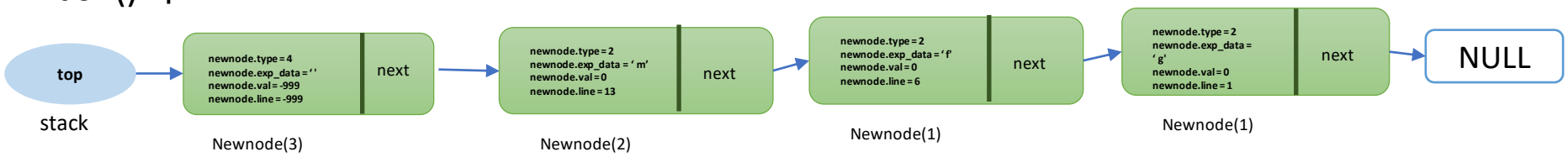
Push

```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```

Node자료형의newnode포인터선언;
만약 newnode는 Node타입크기 만큼메모리를 할당받으며,
newnode가NULL일 경우 ERROR, Couldn't allocate memory... 출력
NULL리턴

newnode.type = 4
newnode.val = -999
newnode.exp_data = ' '
newnode.line = -999
newnode 의 next 는 stck의 top이 된다.
stck의 top은 newnode이다
반환값 stck

Push() 후



int a = 1;

```
//파일의 끝에 도달했는지 여부를 확인
//파일의 끝에 도달하지 못했을 경우에 0 반환
//파일 스트림의 끝을 만날 때까지 반복
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //fget을 통해 function f(int a)을 읽어 line에 저장 .
    fgets(line,4096,filePtr); /* read the file by Line by Line */
    /* scan for /t characters, get rid of them! */
    // whil문 line[k]이  null 이 아닐 시때까지 반복.

    while(line[k]!='₩0')
    {
       //  4-1) 만약 line[k]가 ₩t(탭키)일 시,
     if (line[k]='₩t')
     {
        //line[k]는  ' '으로 바꾼다.
       line[k]=' ';
     }
      //k에 1증가
     k++;
    }
    //5, line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을  ' '로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //begin과 line이 동일하다면
    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }


    //ⓑ
    //  //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //line이 end라면
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }

    //ⓒ begin과 end가 아니면
    else{ ... }


}
```

input 2.sql (읽을 파일)

```
function g(int x)
begin
  (1+2-3+x);
end

function f(int a)
begin
   int b = 1;
   int c = 2;
   ((b*c)+g(a));
end

function main()
begin
   int a = 1;
   int b = 2;
   int c = 3;
   ((2 + f(c) ) * a);
end
```

int

```
// 공백 -> 널0으로 바꿈
firstword=strtok(line," ");

//1.
//int가 firstword이라면
if (!strcmpi("int",firstword))
{
    //(1)-1
    //foundMain ==0
    //foundMain == 0이면 false 0이 아니면 True
    if (foundMain)
    {
        //ttempNode.type 은 1
        tempNode.type=1; /*integer*/

        //자른 문자 다음부터 구분자 또 찾기
        firstword=strtok(NULL," ");

        //tempNode.exp_data의 exp_data = firstword[0]
        tempNode.exp_data=firstword[0];


        //자른 문자 다음부터 구분자  찾기
        firstword=strtok(NULL," ");

        //(2)-1
        /* check for = */
        //firstword가 '='이라면 true
        if (!strcmpi("=",firstword))
        {
            //firstword는 자른 문자 다음부터 구분자  찾기은 값이다.
            firstword=strtok(NULL," ");
        }

        //문자열을 정수 타입 변화 ex) 1
        //tempNode의 val = firstword를 정수 변환
        tempNode.val=atoi(firstword);
        //tempNode의 line은  0
        tempNode.line=0;
        //STACK은 Push(tempNode,STACK)한 값
        STACK=Push(tempNode,STACK);
    }
}
```

firstword=int

firstword=a

firstword= “=“

tempNode.val = 1

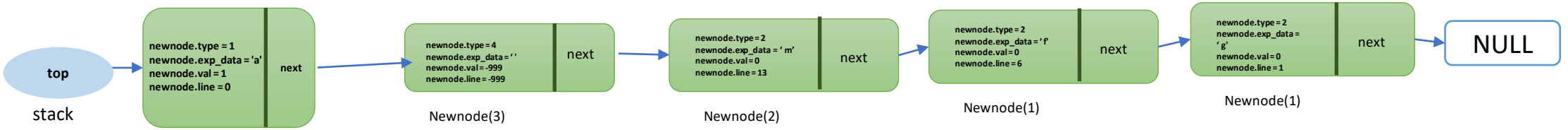
tempNode.line = 0

Push

```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```

Node자료형의newnode포인터선언;
만약 newnode는 Node타입크기 만큼메모리를 할당받으며,
newnode가NULL일 경우 ERROR, Couldn't allocate memory... 출력
NULL리턴

newnode.type = 1
newnode.val = 1
newnode.exp_data = ‘a’
newnode.line = 0
newnode 의 next 는 stck의 top이 된다.
stck의 top은 newnode이다
반환값 stck



```
//파일의 끝에 도달했는지 여부를 확인
//파일의 끝에 도달하지 못했을 경우에 0 반환
//파일 스트림의 끝을 만날 때까지 반복
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //fget을 통해 function f(int a)을 읽어 line에 저장 .
    fgets(line,4096,filePtr); /* read the file by Line by Line */
    /* scan for /t characters, get rid of them! */
    // while문 line[k]이  null 이 아닐 시때까지 반복.

    while(line[k]!='₩0')
    {
      //  4-1) 만약 line[k]가 ₩t(탭키)일 시,
     if (line[k]=='₩t')
     {
       //line[k]는  ' '으로 바꾼다.
      line[k]=' ';
     }
     //k에 1증가
     k++;
    }
    //5, line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을  ' '로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //begin과 line이 동일하다면
    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }


    //ⓑ
    //  //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //line이 end라면
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }

    //ⓒ begin과 end가 아니면
    else{ ... }


}
```

input 2.sql (읽을 파일)

```
function g(int x)
begin
  (1+2-3+x);
end

function f(int a)
begin
   int b = 1;
   int c = 2;
   ((b*c)+g(a));
end

function main()
begin
   int a = 1;
   int b = 2;
   int c = 3;
   ((2 + f(c) ) * a);
end
```

int

```
// 공백 -> 0으로 바꿈
firstword=strtok(line," ");

//1.
//int가 firstword이라면
if (!strcmpi("int",firstword))
{
    //(1)-1
    //foundMain ==0
    //foundMain == 0이면 false 0이 아니면 True
    if (foundMain)
    {
        //ttempNode.type 은 1
        tempNode.type=1; /*integer*/

        //자른 문자 다음부터 구분자 또 찾기
        firstword=strtok(NULL," ");

        //tempNode.exp_data의 exp_data = firstword[0]
        tempNode.exp_data=firstword[0];


        //자른 문자 다음부터 구분자  찾기
        firstword=strtok(NULL," ");

        //(2)-1
        /* check for = */
        //firstword가 '='이라면 true
        if (!strcmpi("=",firstword))
        {
            //firstword는 자른 문자 다음부터 구분자  찾기은 값이다.
            firstword=strtok(NULL," ");
        }

        //문자열을 정수 타입 변화 ex) 1
        //tempNode의 val = firstword를 정수 변환
        tempNode.val=atoi(firstword);
        //tempNode의 line은  0
        tempNode.line=0;
        //STACK은 Push(tempNode,STACK)한 값
        STACK=Push(tempNode,STACK);
    }
}
```

firstword=int

firstword=b

firstword= “=“

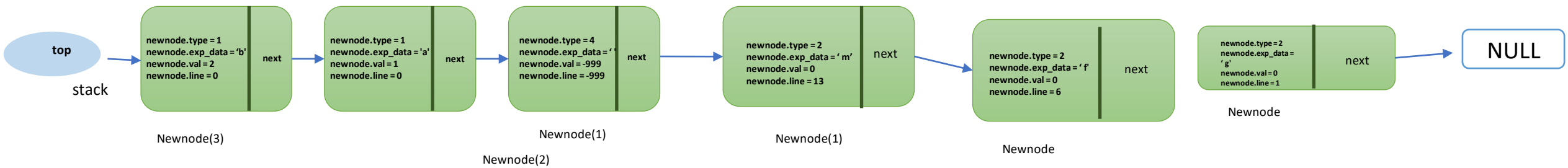
tempNode.val = 2

Push

```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```

Node자료형의newnode포인터선언;
만약 newnode는 Node타입크기 만큼메모리를 할당받으며,
newnode가NULL일 경우 ERROR, Couldn't allocate memory... 출력
NULL리턴

newnode.type = 1
newnode.val = 2
newnode.exp_data = ‘b’
newnode.line = 0
newnode 의 next 는 stck의 top이 된다.
stck의 top은 newnode이다
반환값 stck



```
//파일의 끝에 도달했는지 여부를 확인
//파일의 끝에 도달하지 못했을 경우에 0 반환
//파일 스트림의 끝을 만날 때까지 반복
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //fget을 통해 function f(int a)을 읽어 line에 저장 .
    fgets(line,4096,filePtr); /* read the file by Line by Line */
    /* scan for /t characters, get rid of them! */
    // while문 line[k]이  null 이 아닐 시때까지 반복.

    while(line[k]!='₩0')
    {
       //  4-1) 만약 line[k]가 ₩t(탭키)일 시,
     if (line[k]=='₩t')
     {
        //line[k]는  ' '으로 바꾼다.
       line[k]=' ';
     }
      //k에 1증가
     k++;
    }
    //5, line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을  ' '로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //begin과 line이 동일하다면
    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }


    //ⓑ
    //  //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //line이 end라면
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }

    //ⓒ begin과 end가 아니면
    else{ ... }


}
```

input 2.sql (읽을 파일)

```
function g(int x)
begin
 (1+2-3+x);
end

function f(int a)
begin
  int b = 1;
  int c = 2;
  ((b*c)+g(a));
end

function main()
begin
  int a = 1;
  int b = 2;
  int c = 3;
  ((2 + f(c) ) * a);
end
```

int

```
// 공백 -> \0으로 바꿈
firstword=strtok(line," ");

//1.
//int가 firstword이라면
if (!strcmpi("int",firstword))
{
    //(1)-1
    //foundMain ==0
    //foundMain == 0이면 false 0이 아니면 True
    if (foundMain)
    {
        //ttempNode.type 은 1
        tempNode.type=1; /*integer*/

        //자른 문자 다음부터 구분자 또 찾기
        firstword=strtok(NULL," ");

        //tempNode.exp_data의 exp_data = firstword[0]
        tempNode.exp_data=firstword[0];


        //자른 문자 다음부터 구분자  찾기
        firstword=strtok(NULL," ");

        //(2)-1
        /* check for = */
        //firstword가 '='이라면 true
        if (!strcmpi("=",firstword))
        {
            //firstword는 자른 문자 다음부터 구분자  찾기은 값이다.
            firstword=strtok(NULL," ");
        }

        //문자열을 정수 타입 변화 ex) 1
        //tempNode의 val = firstword를 정수 변환
        tempNode.val=atoi(firstword);
        //tempNode의 line은  0
        tempNode.line=0;
        //STACK은 Push(tempNode,STACK)한 값
        STACK=Push(tempNode,STACK);
    }
}
```

firstword=int

firstword=c

firstword= “=“

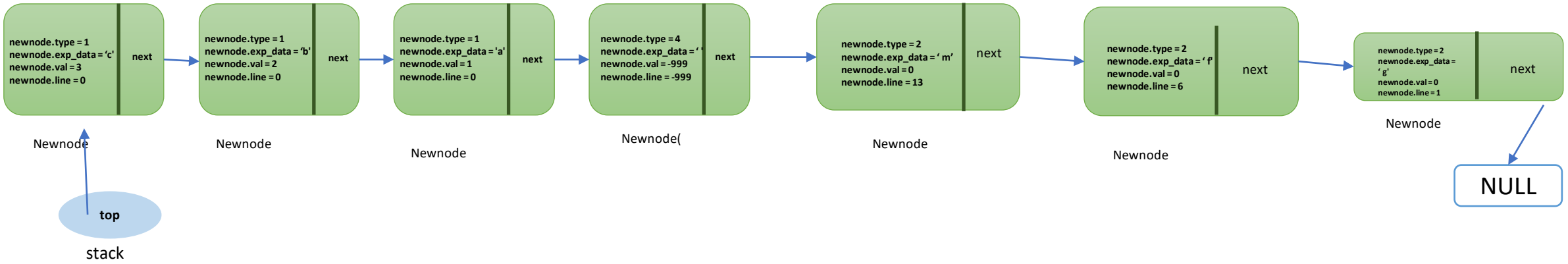
tempNode.val = 3

Push

```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```

Node자료형의newnode포인터선언;
만약 newnode는 Node타입크기 만큼메모리를 할당받으며,
newnode가NULL일 경우 ERROR, Couldn't allocate memory... 출력
NULL리턴

newnode.type = 1
newnode.val = 3
newnode.exp_data = ‘c’
newnode.line = 0
newnode 의 next 는 stck의 top이 된다.
stck의 top은 newnode이다
반환값 stck


newnode.type = 1
newnode.exp_data = 'c'
newnode.val = 3
newnode.line = 0
next
Newnode
top
stack
newnode.type = 1
newnode.exp_data = 'b'
newnode.val = 2
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'a'
newnode.val = 1
newnode.line = 0
next
Newnode
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
Newnode(
newnode.type = 2
newnode.exp_data = ' m'
newnode.val = 0
newnode.line = 13
next
Newnode
newnode.type = 2
newnode.exp_data = ' f'
newnode.val = 0
newnode.line = 6
next
Newnode
newnode.type = 2
newnode.exp_data =
' g'
newnode.val = 0
newnode.line = 1
next
Newnode
NULL

```
//파일 스트림의 끝을 만날 때까지 반복
while (!feof(filePtr))
{

    int k=0;

    //입력 스트림에서 문자열 읽기

    fgets(line,4096, filePtr); /* read the file by Line by Line */
    /* scan for /t characters. get rid of them! */
    while(line[k]!='₩0')
    {
    if (line[k]=='₩t')
    {
       line[k]=' ';
    }

    k++;
    }

    strcpy(lineyedek,line);

    curLine++;
    tempNode.val=-999;
    tempNode.exp_data=' ';
    tempNode.line=-999;
    tempNode.type=-999;



    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }
    else{ ... }


}
```

input 2.sql (읽을 파일)

```
function g(int x)
begin
  (1+2-3+x);
end

function f(int a)
begin
   int b = 1;
   int c = 2;
   ((b*c)+g(a));
end

function main()
begin
   int a = 1;
   int b = 2;
   int c = 3;
   ((2 + f(c) ) * a);
end
```

((2 + f(c)) * a);

firstword[0]= (라면

```
else if (firstword[0]=='(')
{

//foundMain == 0이면 false 0이 아니면 True
if (foundMain)
   {

   int i=0;
   int y=0; //3

   //OpStack + MathStack
   //MathStack.top은 NULL
   MathStack->top=NULL;
   /* now make the postfix calculation */



   //(3)-2
   //lineyedek[i]이 NULL이 아닐때까지 반복
   while(lineyedek[i]!='₩x0')
   {
      //(3)-3-1
      /* evulate the function */
      //숫자라면 true
      if (isdigit(lineyedek[i])){ ... }
       /* ... */

      //(3)-3-2
      //lineyedek[i]이 ')'이라면
      else if (lineyedek[i]==')'){ ... }

      ////(3)-3-3
      //lineyedek[i]이 '+'이거나 lineyedek[i]이 '-' 이거나 lineyedek[i]이 '+' 이거나 lineyedek[i]이 '/'이라면
      else if ((lineyedek[i]=='+') | (lineyedek[i]=='-') | (lineyedek[i]=='+') | (lineyedek[i]=='/')){ ... }

      ////(3)-3-4
      //알파벳 대문자 "A-Z"는 1를 반환,알파벳 소문자 'a-z"는 2를 반환.
      //lineyedek[i] 영어라면
      else if (isalpha(lineyedek[i])>0){ ... }

      //1 증가
      i++;
   }//while문
```

```
/* evulate the function */
//숫자라면 true
if (isdigit(lineyedek[i])) {
   postfix[y]=lineyedek[i];
   y++;

}
 /*      */
```

postfix[0] = 2

(2 + f(c)) * a);

만약 isStackEmpty( ) 0이아니라면

```
////(3)-3-3
//lineyedek[i]이 '+'이거나 lineyedek[i]이 '-' 이거나 lineyedek[i]이 '+' 이거나 lineyedek[i]이 '/'이라면
else if ((lineyedek[i]=='+') | (lineyedek[i]=='-') | (lineyedek[i]=='*') | (lineyedek[i]=='/'))
{
   ////(3)-3-3-1
   /*operators*/
   //00이면 false 10이면 true
   //처음일 때
   if (isStackEmpty(MathStack) != 0 ) //1일때 - stck->top==0
   {
/* if stack empty push the operator to stack */
      //+
      MathStack=PushOp(lineyedek[i],MathStack);
   }
   ////(3)-3-3-2

   //0일때 - stck -top = 00이 아닐때
   //처음이 아닐때
   else { ... }
}
```

ⓐ

ⓑ

PushOp(+, MathStack)

isStackEmpty

```
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)

1. 만약 stck의 top가 0이라면
   1. 반환값 1
2. 아니면 반환값 0

PushOp

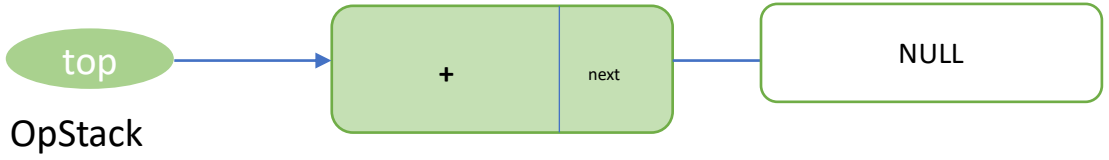
```
OpStack * PushOp(char op,OpStack *opstck)
{
opNode *newnode;
if ((newnode=(opNode*)malloc(sizeof(opNode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->op=op;
newnode->next=opstck->top;
opstck->top=newnode;
return opstck;
}
}
```

OpStack * PushOp(char op,OpStack *opstck)

1. opNode자료형의 newnode포인터선언;
2. 만약 newnode는 opNode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   반환NULL
3. 아니라면,
   1. newnode의 op는 매개변수+이다.
   2. newnode의 next는 opstck의 top이다.
   3. opstck의 top은 newnode이다.

반환값 opstck

PushOp



(2 + f(c)) * a);

```
//lineyedek[i] 영어라면
else if (isalpha(lineyedek[i])>0)
{
    int codeline=0;
    int dummyint=0;
    /*look if it's a variable or function call
    int retVal=0;
    retVal=GetVal(lineyedek[i],&codeline,STACK);
    //int codeline =1

    //-1


    ////(3)-3-4-1
    if ((retVal!=-1) & (retVal!=-999)){ ... }
    ////(3)-3-4-2
    //-1
    else{ ... }
}
```

GetVal (f,&codeline,STACK)

GetVal

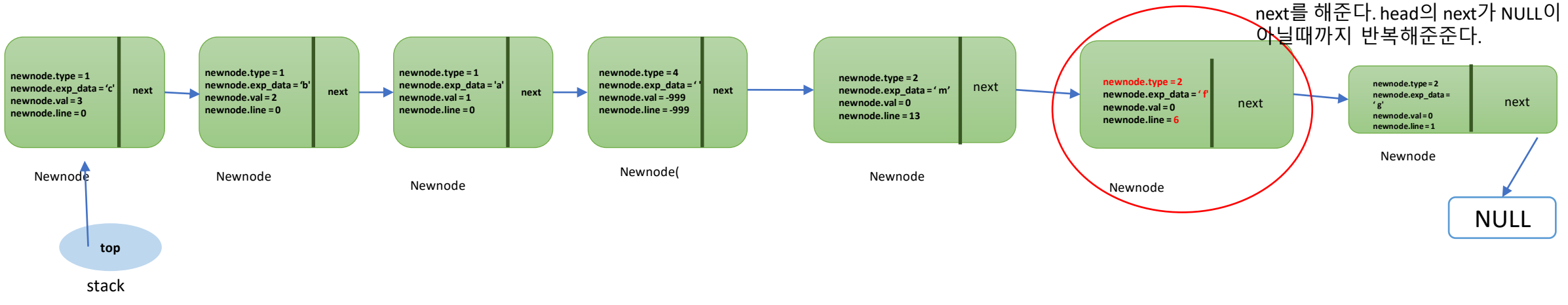
```
int GetVal(char exp_name,int * line,Stack *stck)
{

Node * head;
*line=0;
    if (stck->top == NULL )
    {
        printf("ERROR, empty stack...");
    }
    else
    {
        head=stck->top;
        do
        {
            if ( head->exp_data==exp_name )
            {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }
            }
            else
            {
                head=head->next;
            }
        } while (head->next!=NULL);
        /* check agin once more */
        if (head->exp_data==exp_name)
        {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }


        }
    }

    return -999;
}
```

head->exp_data = f

*line = 6
return -1

GetVal()
자료형인 Node인 포인트 head선언

1. head를 stck의 top으로 둔다.
2. head의exp_data== exp_name
3. 와 같다면
    1. head의 type이 1이라면 head의 val이 반환값이다.
    2. head의 type이 2이라면 line은head의 line이다. -1 반환값
4. head의exp_data==exp_name
5. 와 같은게 없다면 head는 head의 next를 해준다. head의 next가 NULL이 아닐때까지 반복해준준다.



(2 + f(c)) * a);

```
if (LastFunctionReturn==-999)
{

/* if function */
/* add to our system stack that we are making a call to function */
int j;
tempNode.type=3;
tempNode.line=curLine;
STACK=Push(tempNode,STACK);

/* get function's arguments value */
CalingFunctionArgVal=GetVal(lineyedek[i+2],&dummyint,STACK);

fclose(filePtr);
filePtr=fopen(argv[1],"r");
curLine=0;
/* file reversed to start postion */
/* now go codeline lines to go, to the functions line */

codeline olabilir */
for(j=1;j<codeline;j++)
{
   fgets(dummy,4096,filePtr); /*
   curLine++;
}

WillBreak=1;
break;
}
```

1

2

CalingFunctionArgVal =3 (c값)

File을 닫고 읽기모드로 다시연다

curLine 은 0이다

codeline = 6

dummy에 저장

WillBreak=1

push

```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```

Stack * Push(Node sNode,Stack *stck)
1. Node 자료형의 newnode포인터선언;
2. 만약 newnode는 Node타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   return NULL
3.아니라면
   newnode의 type은 3 된다.
   newnode 의 val은 0 이된다.
   newnode 의 exp_data은 ‘’이된다.
   newnode 의 line 은 이 된다.
   newnode 의 next 는 stck의 top이 된다.
   stck의 top은 newnode이다
반환값 stck

head->exp_data = c

head 포인트 생성.
head는 stck의 top을 가리킨다.
if(head->exp_data = ‘c’)이라면서
1. head의 type이 1이라면
2. head의 val 리턴
head의 next가 NULL이 아닐때까지 반복

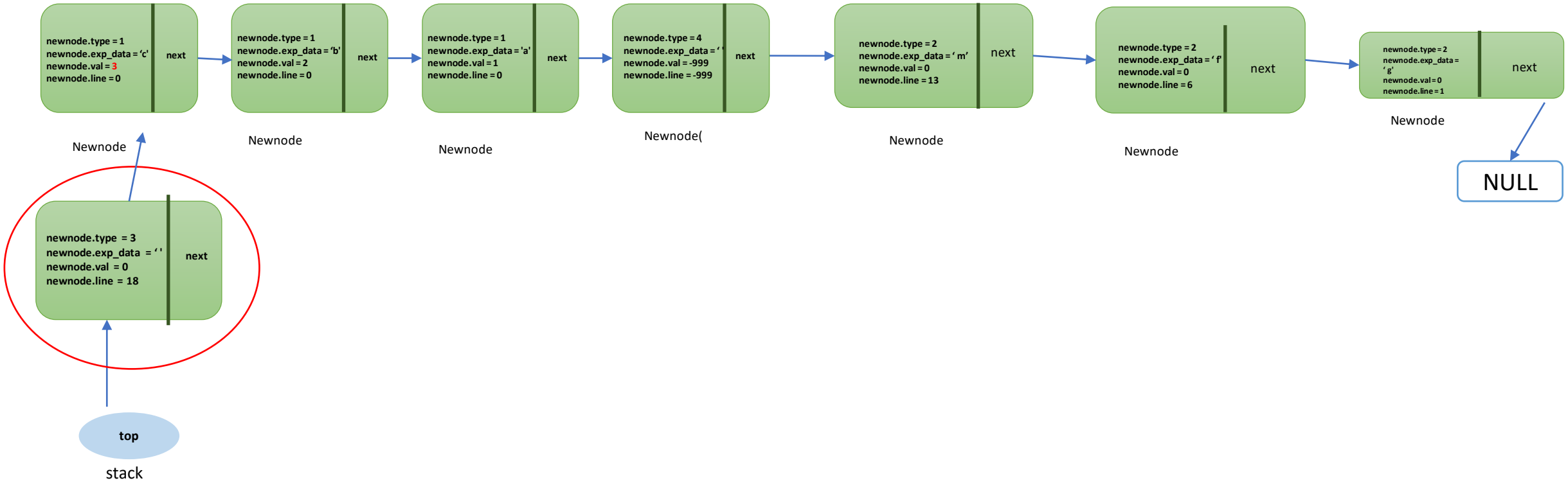
GetVal

```
int GetVal(char exp_name,int * line,Stack *stck)
{

Node * head;
*line=0;
    if (stck->top == NULL )
    {
        printf("ERROR, empty stack...");
    }
    else
    {
        head=stck->top;
        do
        {
            if ( head->exp_data==exp_name )
            {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }

            }
            else
            {
                head=head->next;
            }
        } while (head->next!=NULL);
        /* check agin once more */
        if (head->exp_data==exp_name)
        {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }


        }
    }

    return -999;
}
```

for문
1. j는 1부터 j가 6보다 작을때까지 1씩증가하며 반복
2. filePtr의파일을 최대 4095수까지읽으며,dummy배열에 저장.
3. curLine 1증가

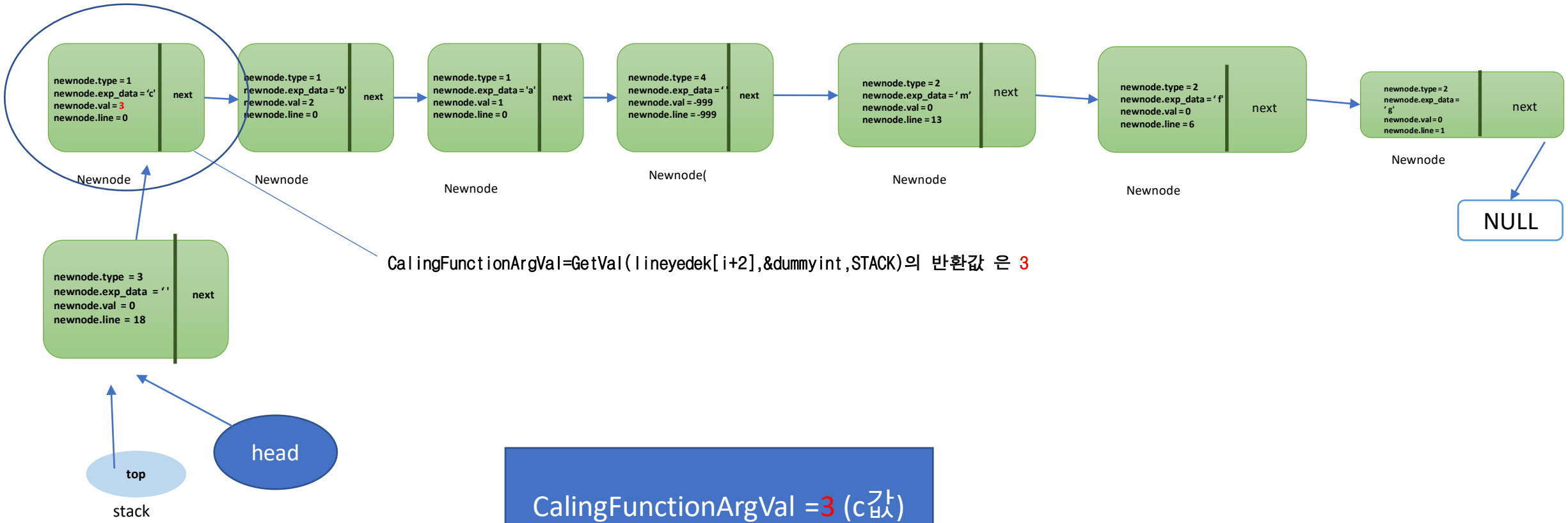
1

push()


newnode.type = 1
newnode.exp_data = 'c'
newnode.val = 3
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'b'
newnode.val = 2
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'a'
newnode.val = 1
newnode.line = 0
next
Newnode
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
Newnode(
newnode.type = 2
newnode.exp_data = ' m'
newnode.val = 0
newnode.line = 13
next
Newnode
newnode.type = 2
newnode.exp_data = ' f'
newnode.val = 0
newnode.line = 6
next
Newnode
newnode.type = 2
newnode.exp_data = ' g'
newnode.val = 0
newnode.line = 1
next
Newnode
NULL
newnode.type = 3
newnode.exp_data = ' '
newnode.val = 0
newnode.line = 18
next
top
stack

2

GetVal()



CalingFunctionArgVal =3 (c값)

```
    if (LastFunctionReturn==-999)
    {
    /* if function */
    /* add to our system stack that we are making a call to function */
    int j;
    tempNode.type=3;
    tempNode.line=curLine;
    STACK=Push(tempNode,STACK);

    /* get function's arguments value */
    CalingFunctionArgVal=GetVal(lineyedek[i+2],&dummyint,STACK);



    fclose(filePtr);
    filePtr=fopen(argv[1],"r");
    curLine=0;
    /* file reversed to start postion */
    /* now go codeline lines to go, to the functions line */


codeline olabilir */
    for(j=1;j<codeline;j++)
    {
        fgets(dummy,4096,filePtr); /* read the file by Line by Line */
        curLine++;
    }

    WillBreak=1;
    break;
    }
```

break를 만나 whil문을 빠져나간다.

```
//lineyedek[i]이 NULL이 아닐때까지 반복
while(lineyedek[i]!='₩x0')
{
    //(3)-3-1
    /* evulate the function */
    //숫자라면 true
    if (isdigit(lineyedek[i])) {
        postfix[y]=lineyedek[i];

        y++;  //1증가

    }
     /* ... */

    //(3)-3-2
    //lineyedek[i]이 ')'이라면
    else if (lineyedek[i]==')')
    {
        //(3)-3-2-1
        ////00이면 true 10이면 false
        //0일때
        if (!isStackEmpty(MathStack) != 0 ) //- OpStack->top==00| 아닐 때
        {
            postfix[y]=PopOp(MathStack);
            y++; // 1증가
        }
    }

    ////(3)-3-3
    //lineyedek[i]이 '+'이거나 lineyedek[i]이 '-' 이거나 lineyedek[i]이 '+' 이거나 lineyedek[i]이 '/'이라면
    else if ((lineyedek[i]=='+') | (lineyedek[i]=='-') | (lineyedek[i]=='*') | (lineyedek[i]=='/')){ ... }

    ////(3)-3-4
    //알파벳 대문자 "A-Z"는 1을 반환.알파벳 소문자 'a-z'는 2를 반환.
    //lineyedek[i] 영어라면
    else if (isalpha(lineyedek[i])>0){ ... }

    //1 증가
    i++;
}//while문


//WillBreak이 0이라면
if (WillBreak==0){ ... }
WillBreak=0;
}
```

다시 reset

function f(int a)

```
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //fget을 통해 function f(int a)을 읽어 line에 저장 .
    fgets(line,4096,filePtr); /* read the file by Line by Line */
    /* scan for /t characters. get rid of them! */

    // whil문 line[k]이  null 이 아닐 시때까지 반복.
    while(line[k]!='₩0')
    {
       //  4-1) 만약 line[k]가 ₩t(탭키)일 시,
     if (line[k]=='₩t')
     {
       //line[k]는  ' '으로 바꾼다.
      line[k]=' ';
     }
      //k에 1증가
     k++;
    }
    //5. line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을  ' '로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  둘이 동일하면 1(true) 반환하고 동일하지 않으면 0(false)
    //begin과 line이 동일하다면
    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }


    //ⓑ
    //  //strcmpi :  둘이 동일하면 1(true) 반환하고 동일하지 않으면 0(false)
    //line이 end라면
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }

    //ⓒ begin과 end가 아니면
    else{ ... }
```

curLine 은 6이다

input 2.sql (읽을 파일)

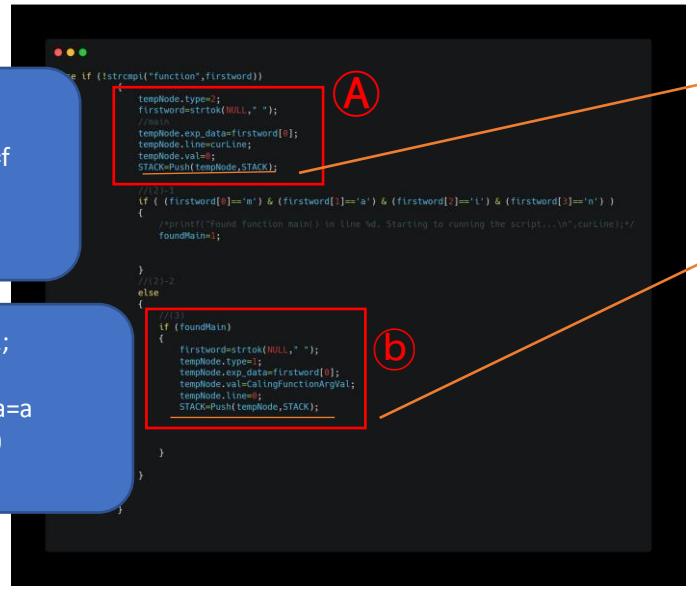
```
function g(int x)
begin
 (1+2-3+x);
end

function f(int a)
begin
  int b = 1;
  int c = 2;
  ((b*c)+g(a));
end

function main()
begin
  int a = 1;
  int b = 2;
  int c = 3;
  ((2 + f(c) ) * a);
end
```

다시 reset

function f(int a)

tempNode.type=2;
firstword= f(int
tempNode.exp_data=f
tempNode.line=6
tempNode.val=0

tempNode.type=1;
firstword= x)
tempNode.exp_data=a
tempNode.line=0
tempNode.val=3;



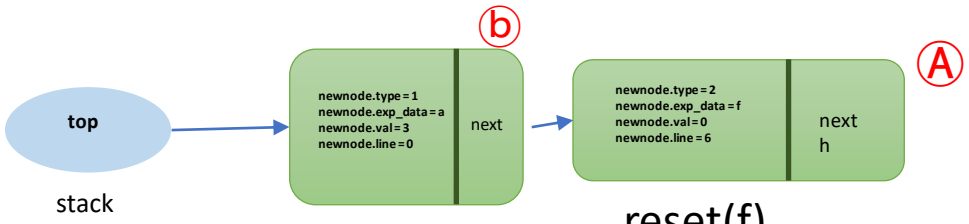
function

Push

```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```

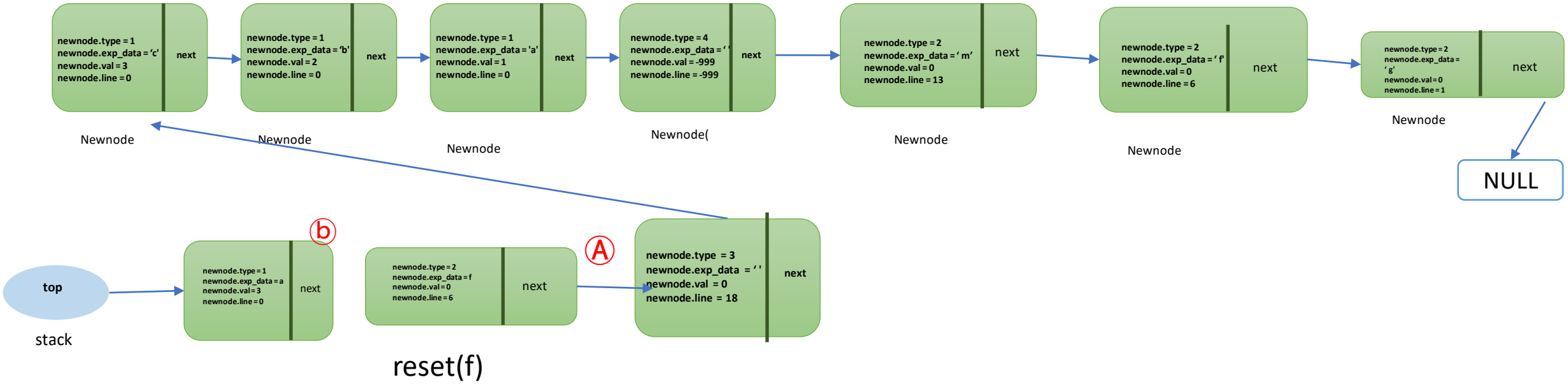
Node자료형의newnode포인터선언;
만약 newnode는 Node타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우 ERROR, Couldn't allocate memory... 출력
NULL리턴

newnode.type = 1
newnode.val = 4
newnode.exp_data = ' a'
newnode.line = 0
newnode 의 next 는 stck의 top이 된다.
stck의 top은 newnode이다
반환값 stck



reset(f)

function f 파일을 새로 다시 시작한다.
function f 에서의 새 노드를
reset(f)라고 표기하였다.


newnode.type = 1
newnode.exp_data = 'c'
newnode.val = 3
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'b'
newnode.val = 2
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'a'
newnode.val = 1
newnode.line = 0
next
Newnode
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
Newnode(
newnode.type = 2
newnode.exp_data = ' m'
newnode.val = 0
newnode.line = 13
next
Newnode
newnode.type = 2
newnode.exp_data = ' f'
newnode.val = 0
newnode.line = 6
next
Newnode
newnode.type = 2
newnode.exp_data = ' g'
newnode.val = 0
newnode.line = 1
next
Newnode
NULL
b
top
stack
newnode.type = 1
newnode.exp_data = a
newnode.val = 3
newnode.line = 0
next
newnode.type = 2
newnode.exp_data = f
newnode.val = 0
newnode.line = 6
next
A
newnode.type = 3
newnode.exp_data = ' '
newnode.val = 0
newnode.line = 18
next
reset(f)

파일반복

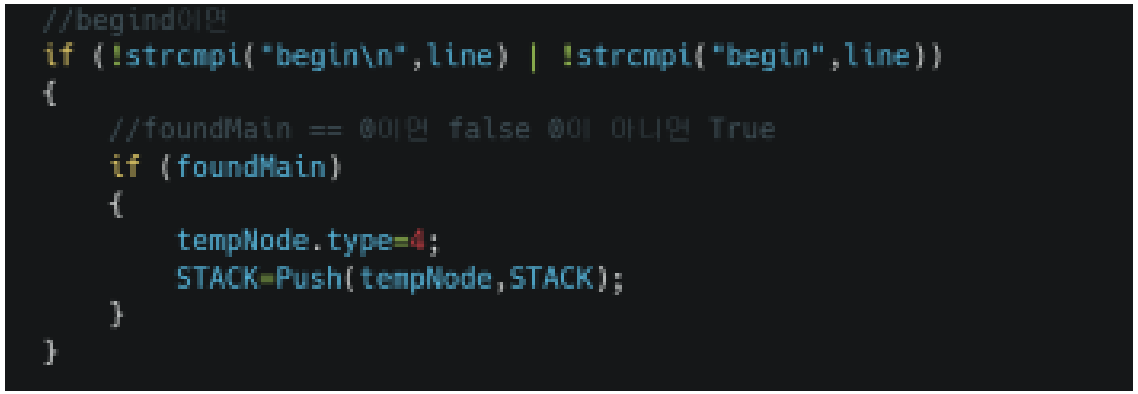
```
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //fget을 통해 function f(int a)을 읽어 line에 저장 .
    fgets(line,4096,filePtr); /* read the file by Line by Line */
    /* scan for /t characters. get rid of them! */

    // whil문 line[k]이  null 이 아닐 시때까지 반복.
    while(line[k]!='₩0')
    {
       //  4-1) 만약 line[k]가 ₩t(탭키)일 시,
     if (line[k]=='₩t')
     {
       //line[k]는  ' '으로 바꾼다.
      line[k]=' ';
     }
      //k에 1증가
     k++;
    }
    //5. line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을  ' '로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  둘이 동일하면 1(true) 반환하고 동일하지 않으면 0(false)
    //begin과 line이 동일하다면
    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }


    //ⓑ
    //  //strcmpi :  둘이 동일하면 1(true) 반환하고 동일하지 않으면 0(false)
    //line이 end라면
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }

    //ⓒ begin과 end가 아니면
    else{ ... }
```

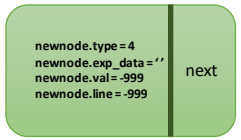
input 2.sql (읽을 파일)

```
function g(int x)
begin
  (1+2-3+x);
end

function f(int a)
begin
  int b = 1;
  int c = 2;
  ((b*c)+g(a));
end

function main()
begin
  int a = 1;
  int b = 2;
  int c = 3;
  ((2 + f(c) ) * a);
end
```

begin

begin이라면



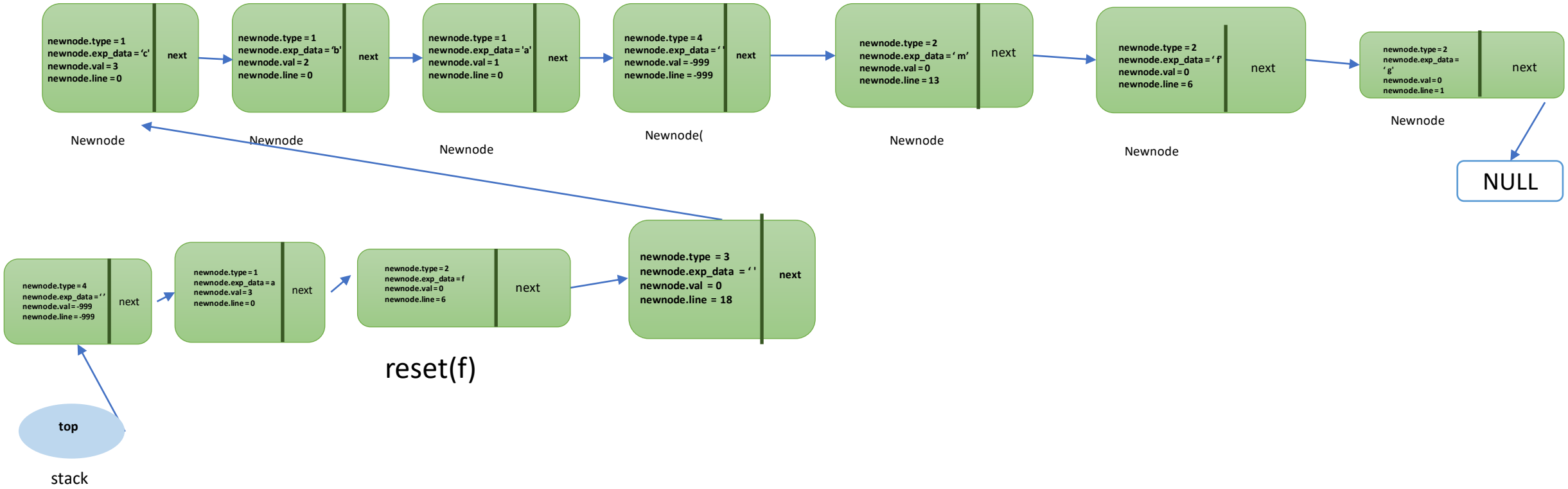


Push

```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```

Node자료형의newnode포인터선언;
만약 newnode는 Node타입크기 만큼메모리를 할당받으며,
newnode가NULL일 경우 ERROR, Couldn't allocate memory... 출력
NULL리턴

newnode.type = 4
newnode.val = -999
newnode.exp_data = ‘’
newnode.line = -999
newnode 의 next 는 stck의 top이 된다.
stck의 top은 newnode이다
반환값 stck


newnode.type = 1
newnode.exp_data = 'c'
newnode.val = 3
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'b'
newnode.val = 2
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'a'
newnode.val = 1
newnode.line = 0
next
Newnode
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
Newnode(
newnode.type = 2
newnode.exp_data = ' m'
newnode.val = 0
newnode.line = 13
next
Newnode
newnode.type = 2
newnode.exp_data = ' f'
newnode.val = 0
newnode.line = 6
next
Newnode
newnode.type = 2
newnode.exp_data =
' g'
newnode.val = 0
newnode.line = 1
next
Newnode
NULL
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
newnode.type = 1
newnode.exp_data = a
newnode.val = 3
newnode.line = 0
next
newnode.type = 2
newnode.exp_data = f
newnode.val = 0
newnode.line = 6
next
newnode.type = 3
newnode.exp_data = ' '
newnode.val = 0
newnode.line = 18
next
top
stack
reset(f)

파일반복

```
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //fget을 통해 function f(int a)을 읽어 line에 저장 .
    fgets(line,4096,filePtr); /* read the file by Line by Line */
    /* scan for /t characters. get rid of them! */

    // whil문 line[k]이  null 이 아닐 시때까지 반복.
    while(line[k]!='₩0')
    {
       //  4-1) 만약 line[k]가 ₩t(탭키)일 시,
     if (line[k]=='₩t')
     {
       //line[k]는  ' '으로 바꾼다.
      line[k]=' ';
     }
      //k에 1증가
     k++;
    }
    //5. line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을  ' '로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  둘이 동일하면 1(true) 반환하고 동일하지 않으면 0(false)
    //begin과 line이 동일하다면
    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }


    //ⓑ
    //  //strcmpi :  둘이 동일하면 1(true) 반환하고 동일하지 않으면 0(false)
    //line이 end라면
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }

    //ⓒ begin과 end가 아니면
    else{ ... }
```

input 2.sql (읽을 파일)

```
function g(int x)
begin
 (1+2-3+x);
end

function f(int a)
begin
  int b = 1;
  int c = 2;
  ((b*c)+g(a));
end

function main()
begin
  int a = 1;
  int b = 2;
  int c = 3;
  ((2 + f(c) ) * a);
end
```

int

```
// 공백 -> 0으로 바꿈
firstword=strtok(line," ");

//1.
//int가 firstword이라면
if (!strcmpi("int",firstword))
{
    //(1)-1
    //foundMain ==0
    //foundMain == 0이면 false 0이 아니면 True
    if (foundMain)
    {
        //ttempNode.type 은 1
        tempNode.type=1; /*integer*/

        //자른 문자 다음부터 구분자 또 찾기
        firstword=strtok(NULL," ");

        //tempNode.exp_data의 exp_data = firstword[0]
        tempNode.exp_data=firstword[0];


        //자른 문자 다음부터 구분자  찾기
        firstword=strtok(NULL," ");

        //(2)-1
        /* check for = */
        //firstword가 '='이라면 true
        if (!strcmpi("=",firstword))
        {
            //firstword는 자른 문자 다음부터 구분자  찾기은 값이다.
            firstword=strtok(NULL," ");
        }

        //문자열을 정수 타입 변화 ex) 1
        //tempNode의 val = firstword를 정수 변환
        tempNode.val=atoi(firstword);
        //tempNode의 line은  0
        tempNode.line=0;
        //STACK은 Push(tempNode,STACK)한 값
        STACK=Push(tempNode,STACK);
    }
}
```

firstword=int

firstword=c

firstword= “=“

tempNode.val = 2

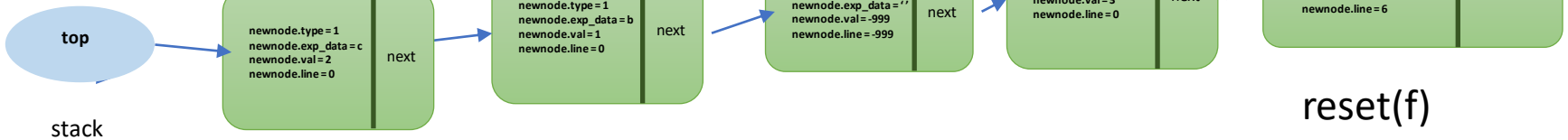
tempNode.line = 0

Push

```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```

Node자료형의newnode포인터선언;
만약 newnode는 Node타입크기 만큼메모리를 할당받으며,
newnode가NULL일 경우 ERROR, Couldn't allocate memory... 출력
NULL리턴

newnode.type = 1
newnode.val = 2
newnode.exp_data = ‘c’
newnode.line = 0
newnode 의 next 는 stck의 top이 된다.
stck의 top은 newnode이다
반환값 stck



stack

reset(f)

```
//파일의 끝에 도달했는지 여부를 확인
//파일의 끝에 도달하지 못했을 경우에 0 반환
//파일 스트림의 끝을 만날 때까지 반복
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //fget을 통해 function f(int a)을 읽어 line에 저장 .
    fgets(line,4096,filePtr); /* read the file by Line by Line */
    /* scan for /t characters, get rid of them! */
    // while문 line[k]이  null 이 아닐 시때까지 반복.

    while(line[k]!='₩0')
    {
       //  4-1) 만약 line[k]가 ₩t(탭키)일 시,
     if (line[k]=='₩t')
     {
        //line[k]는  ' '으로 바꾼다.
       line[k]=' ';
     }
      //k에 1증가
     k++;
    }
    //5, line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을  ' '로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //begin과 line이 동일하다면
    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }


    //ⓑ
    //  //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //line이 end라면
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }

    //ⓒ begin과 end가 아니면
    else{ ... }


}
```

input 2.sql (읽을 파일)

```
function g(int x)
begin
  (1+2-3+x);
end

function f(int a)
begin
   int b = 1;
   int c = 2;
   ((b*c)+g(a));
end

function main()
begin
   int a = 1;
   int b = 2;
   int c = 3;
   ((2 + f(c) ) * a);
end
```

((b*c)+g(a));

```
//lineyedek[i] 영어라면
else if (isalpha(lineyedek[i])>0)
{
    int codeline=0;
    int dummyint=0;
    /*look if it's a variable or function call
    int retVal=0;
    retVal=GetVal(lineyedek[i],&codeline,STACK);
    //int codeline =1

    //-1


    ////(3)-3-4-1
    if ((retVal!=-1) & (retVal!=-999)){ ... }
    ////(3)-3-4-2
    //-1
    else{ ... }
}
```

GetVal (b,&codeline,STACK)

GetVal

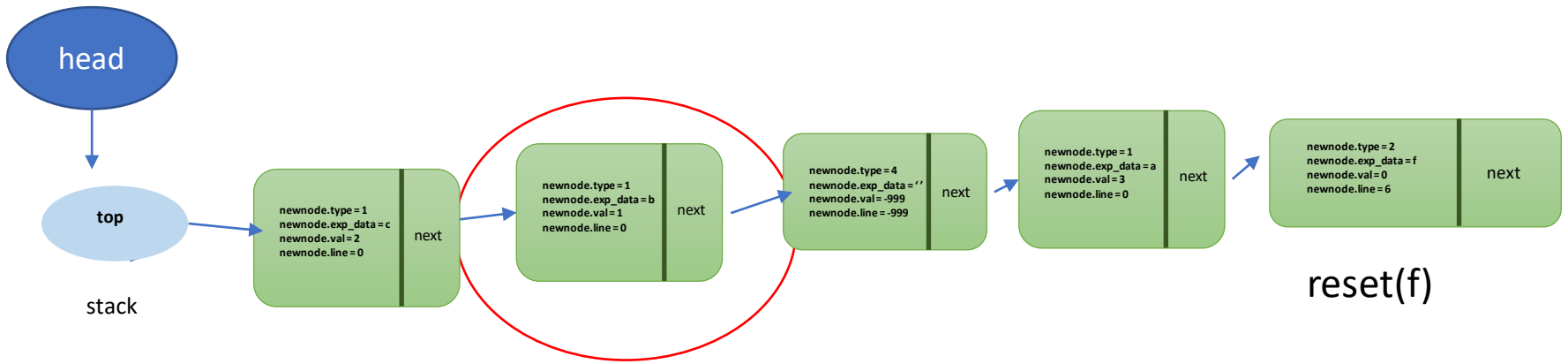
```
int GetVal(char exp_name,int * line,Stack *stck)
{

Node * head;
*line=0;
    if (stck->top == NULL )
    {
        printf("ERROR, empty stack...");
    }
    else
    {
        head=stck->top;
        do
        {
            if ( head->exp_data==exp_name )
            {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }

            }
            else
            {
                head=head->next;
            }
        } while (head->next!=NULL);
        /* check agin once more */
        if (head->exp_data==exp_name)
        {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }


        }
    }

    return -999;
}
```

head->exp_data = b

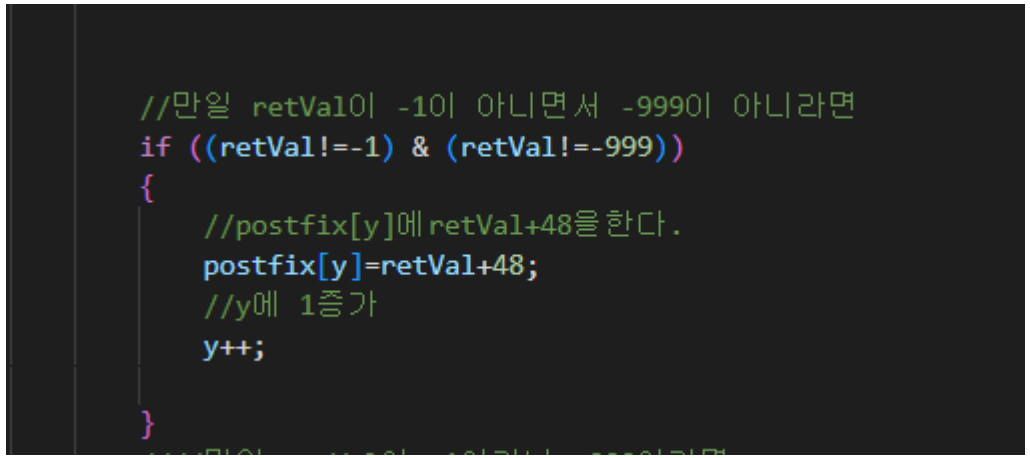
return 1

GetVal()
자료형인 Node인 포인트 head선언

1. head를 stck의 top으로 둔다.
2. head의exp_data==’b’와 같다면
    1. head의 type이 1이라면
       head의 val이 반환값이다.
    2. head의 type이 2이라면
       line은head의 line이다.
       -1 반환값
3. head의exp_data==’b’와 같은게 없다면 head는 head의 next를 해준다. head의 next가 NULL이 아닐때까지 반복해준준다.



((b*c)+g(a));



```
//만일 retVal이 -1이 아니면서 -999이 아니라면
if ((retVal!=-1) & (retVal!=-999))
{
    //postfix[y]에 retVal+48을 한다.
    postfix[y]=retVal+48;
    //y에 1증가
    y++;

}
```

retVal = 1

postfix[0] = 49

((b*c)+g(a));

```
////(3)-3-3
//lineyedek[i]이 '+'이거나 lineyedek[i]이 '-' 이거나 lineyedek[i]이 '+' 이거나 lineyedek[i]이 '/'이라면
else if ((lineyedek[i]=='+') | (lineyedek[i]=='-') | (lineyedek[i]=='*') | (lineyedek[i]=='/'))
{
   ////(3)-3-3-1
   /*operators*/
   //00이면 false 10이면 true
   //처음일 때
   if (isStackEmpty(MathStack) != 0 ) //1일때 - stck->top==0
   {
/* if stack empty push the operator to stack */
      //+
      MathStack=PushOp(lineyedek[i],MathStack);
   }
   ////(3)-3-3-2

   //0일때 - stck -top = 00이 아닐때
   //처음이 아닐때
   else { ... }
}
```

ⓐ

ⓑ

PushOp(*, MathStack)

isStackEmpty

```
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)

1. 만약 stck의 top가 0이라면
   1. 반환값 1
2. 아니면 반환값 0

PushOp

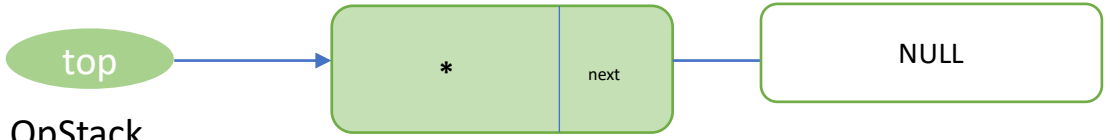
```
OpStack * PushOp(char op,OpStack *opstck)
{
opNode *newnode;
if ((newnode=(opNode*)malloc(sizeof(opNode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->op=op;
newnode->next=opstck->top;
opstck->top=newnode;
return opstck;
}
}
```

OpStack * PushOp(char op,OpStack *opstck)

1. opNode자료형의 newnode포인터선언;
2. 만약 newnode는 opNode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   반환NULL
3. 아니라면,
   1. newnode의 op는 매개변수+이다.
   2. newnode의 next는 opstck의 top이다.
   3. opstck의 top은 newnode이다.

반환값 opstck

PushOp



OpStack
MathStack

((b*c)+g(a));

```
//lineyedek[i] 영어라면
else if (isalpha(lineyedek[i])>0)
{
    int codeline=0;
    int dummyint=0;
    /*look if it's a variable or function call
    int retVal=0;
    retVal=GetVal(lineyedek[i],&codeline,STACK);
    //int codeline =1

    //-1


    ////(3)-3-4-1
    if ((retVal!=-1) & (retVal!=-999)){ ... }
    ////(3)-3-4-2
    //-1
    else{ ... }
}
```

GetVal (c,&codeline,STACK)

GetVal

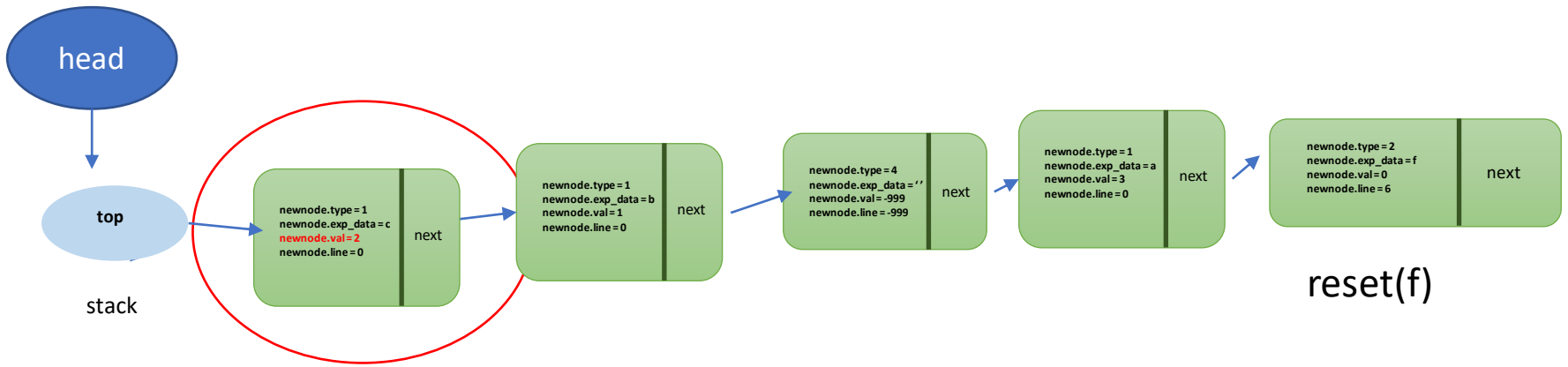
```
int GetVal(char exp_name,int * line,Stack *stck)
{

Node * head;
*line=0;
    if (stck->top == NULL )
    {
        printf("ERROR, empty stack...");
    }
    else
    {
        head=stck->top;
        do
        {
            if ( head->exp_data==exp_name )
            {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }

            }
            else
            {
                head=head->next;
            }
        } while (head->next!=NULL);
        /* check again once more */
        if (head->exp_data==exp_name)
        {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }


        }
    }

    return -999;
}
```

head->exp_data = c

return 2

GetVal()
자료형인 Node인 포인트 head선언

1. head를 stck의 top으로 둔다.
2. head의exp_data==’c’와 같다면
    1. head의 type이 1이라면
       head의 val이 반환값이다.
    2. head의 type이 2이라면
       line은head의 line이다.
       -1 반환값
3. head의exp_data==’c’와 같은게 없다면 head는 head의 next를 해준다. head의 next가 NULL이 아닐때까지 반복해준준다.



((b*c)+g(a));

```
//만일 retVal이 -1이 아니면서 -999이 아니라면
if ((retVal!=-1) & (retVal!=-999))
{
    //postfix[y]에 retVal+48을 한다.
    postfix[y]=retVal+48;
    //y에 1증가
    y++;

}
```

retVal = 2

postfix[1] = 50

((b*c)+g(a));

세분화2-4-1-1 (begin과 end가 아니면) - '('라면 세분화 ⓐ

')'이라면

```
else if (lineyedek[i]==')')
                {
                    //(3)-3-2-1
                    ////0이면 true 1이면 false

                    if (!isStackEmpty(MathStack) != 0 ) //0일때 - OpStack-
>top==0이 아닐 때
                    {
                        // y = 0
                        //Null과 0은 같다
                        postfix[y]=PopOp(MathStack); //
                        y++;
                    }
                }
```

Postfix[2] =‘+’ Postfix[]

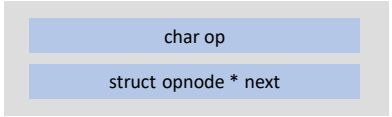
49	50	*

isStackEmpty

```
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)

1. 만약 stck의 top가 0이라면
   1. 반환값 1
2. 아니면 반환값 0



opNode

```
char PopOp(OpStack *opstck)
{
opNode *temp;
char op;
if (opstck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
op=opstck->top->op;
temp=opstck->top;
opstck->top=opstck->top->next;
free(temp);
return op;
}
return NULL;
}
```

PopOp(opstck)

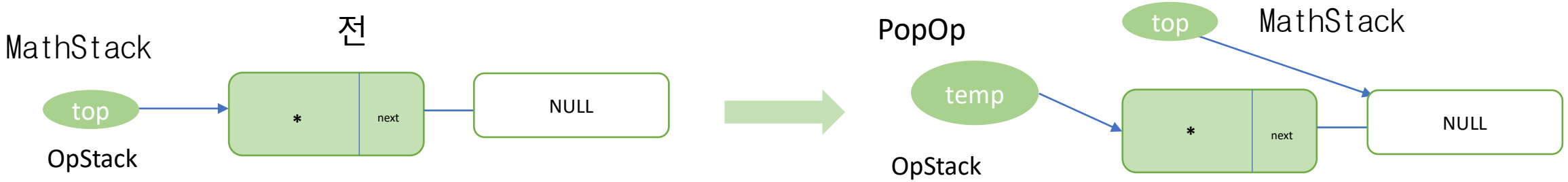
1. opstck의 top이 NULL이면 “Error, empty stack…” 라고 하고 return null;

opstck의 top이 NULL이 아니면,

1. op =*(opstck의 top의 op)를 가리킨다.
2. temp는 opstck의 top이다.
3. opstck의 top은opstck의 top의 next이다.
4. temp를 메모리 해체를한다.
5. return은 *

Op = +

메모리 해제



((b*c)+g(a));

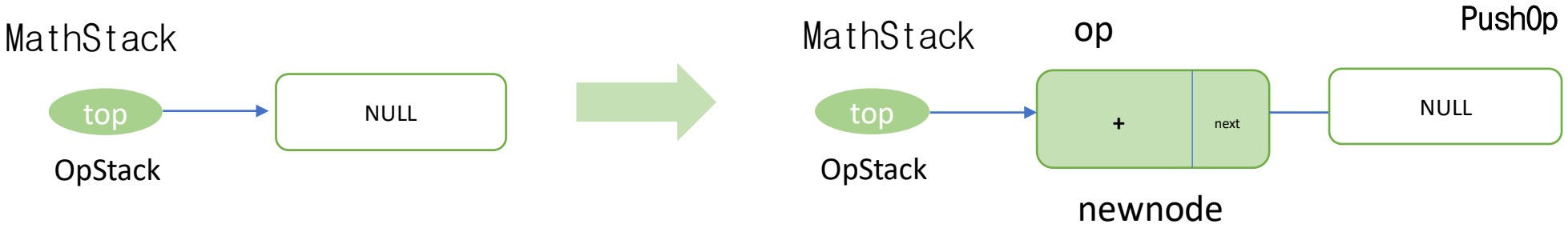
```
    else if ((lineyedek[i]=='+') | (lineyedek[i]=='-') | (lineyedek[i]=='*') |
(lineyedek[i]=='/'))
                {
                    ////(3)-3-3-1
                    /*operators*/
                    //0이면 false 1이면 true
                    //처음일 때
                    if (isStackEmpty(MathStack) != 0 ) //1일때 - stck->top==0
                    {
                /* if stack empty push the operator to stack */

                    MathStack=PushOp(lineyedek[i],MathStack);
                    }
                    ////(3)-3-
```

PsushOp('+' , MathStack)

```
OpStack * PushOp(char op,OpStack *opstck)
{
opNode *newnode;
if ((newnode=(opNode*)malloc(sizeof(opNode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->op=op;
newnode->next=opstck->top;
opstck->top=newnode;
return opstck;
}
}
```

PushOp()
만약 opNode의 구조체 크기가 Null일 경우,
"ERROR, Couldn't allocate memory..."출력 후 NULL 리턴

아니라면,
1. +는 새로운 노드의 op가 된다.
2. opstck->top은 새로운 노드의 next이다
3. newnode는 Opstck의 top은 가리킨다.
return opstck;



((b*c)+g(a));

```
else if (isalpha(lineyedek[i])>0)
                {
                    int codeline=0;
                    int dummyint=0;
                    /*look if it's a v
                    int retVal=0;
                    retVal=GetVal(lineyedek[i],&codeline,STACK);
```

a,0,stack

①

②

retVal = 4

```
////(3)-3-4-1
if ((retVal!=-1) & (retVal!=-999))
{
    /* if variable */
    postfix[y]=retVal+48; /* in asci
    y++; //1

}
```

Postfix[3] =4+48

head

retVal은 -1이 아니며 -999가 아니기에
postfix[3] = 4+48;을 한 후
Y값에 1을 더한다.

GetVal()
자료형인 Node인 포인트 head선언
1. head를 stck의 top으로 둔다.
2. head의exp_data==' g'와 같다면
   1. head의 type이 1이라면 head의 val이 반환값이다.
   2. head의 type이 2이라면
      line은head의 line이다.
      -1 반환값
3. head의exp_data==' g'와 같은게 없다면 head는 head의 next를 해준다. head의 next가 NULL이 아닐때까지 반복해준준다.

head->next가 Null일때까지 반복했는데 없다면 -999리턴

GetVal

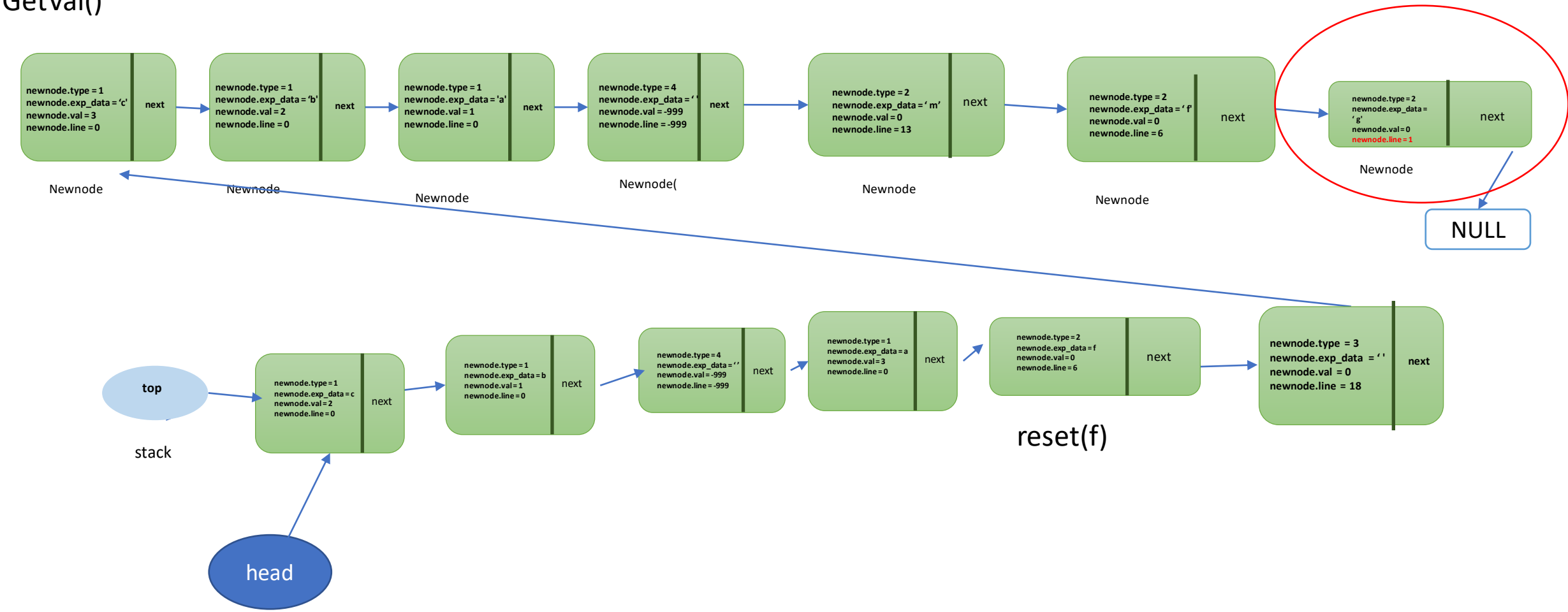
```
head=stck->top;
do
{
    if ( head->exp_data==exp_name )
    {

        if (head->type==1)
        {
            /* return the variables value */
            return head->val;
        }
        else if (head->type==2)
        {
            *line=head->line;
            return -1;
            /* it's a function so return -1 */
        }

    }
    else
    {
        head=head->next;
    }
} while (head->next!=NULL);
/* check agin once more */
```

Return -1

codeline =1

```
    /* check agin once more */
    if (head->exp_data==exp_name)
    {

        if (head->type==1)
        {
            /* return the variables value */
            return head->val;
        }
        else if (head->type==2)
        {
            *line=head->line;
            return -1;
            /* it's a function so return -1 */
        }

    }
}

return -999;
}
```

GetVal()


newnode.type = 1
newnode.exp_data = 'c'
newnode.val = 3
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'b'
newnode.val = 2
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'a'
newnode.val = 1
newnode.line = 0
next
Newnode
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
Newnode(
newnode.type = 2
newnode.exp_data = ' m'
newnode.val = 0
newnode.line = 13
next
Newnode
newnode.type = 2
newnode.exp_data = ' f'
newnode.val = 0
newnode.line = 6
next
Newnode
newnode.type = 2
newnode.exp_data = ' g'
newnode.val = 0
newnode.line = 1
next
Newnode
NULL
top
stack
head
newnode.type = 1
newnode.exp_data = c
newnode.val = 2
newnode.line = 0
next
newnode.type = 1
newnode.exp_data = b
newnode.val = 1
newnode.line = 0
next
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
newnode.type = 1
newnode.exp_data = a
newnode.val = 3
newnode.line = 0
next
newnode.type = 2
newnode.exp_data = f
newnode.val = 0
newnode.line = 6
next
reset(f)
newnode.type = 3
newnode.exp_data = ' '
newnode.val = 0
newnode.line = 18
next

((b*c)+g(a));

```
if (LastFunctionReturn==-999)
{

/* if function */
/* add to our system stack that we are making a call to function */
int j;
tempNode.type=3;
tempNode.line=curLine;
STACK=Push(tempNode,STACK);

/* get function's arguments value */
CalingFunctionArgVal=GetVal(lineyedek[i+2],&dummyint,STACK);

fclose(filePtr);
filePtr=fopen(argv[1],"r");
curLine=0;
/* file reversed to start postion */
/* now go codeline lines to go, to the functions line */


codeline olabilir */
for(j=1;j<codeline;j++)
{
   fgets(dummy,4096,filePtr); /*
   curLine++;
}

WillBreak=1;
break;
}
```

1

2

CalingFunctionArgVal =1 (a값)

File을 닫고 읽기모드로 다시연다

curLine 은 0이다

Dummy에 저장

WillBreak=1

push

```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```

Push(sNode,*stck)
새로운 노드의 타입은 3이되며,
새로운노드의 val은 0,새로운 노드의
exp_data = ‘‘이며,

getVal()

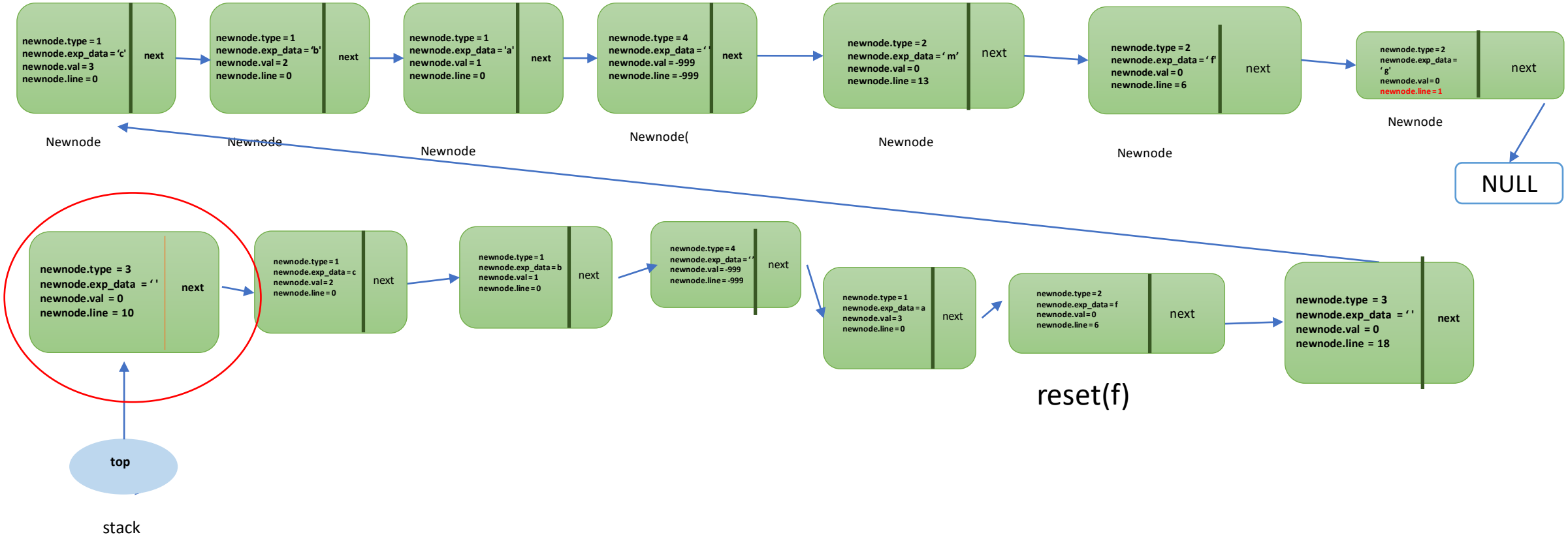
```
int GetVal(char exp_name,int * line,Stack *stck)
{

Node * head;
*line=0;
    if (stck->top == NULL )
    {
        printf("ERROR, empty stack...");
    }
    else
    {
        head=stck->top;
        do
        {
            if ( head->exp_data==exp_name )
            {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }
```

head->exp_data = a

if(head->exp_data = a)이라면
head의 type이 1이라면
head의 val 리턴

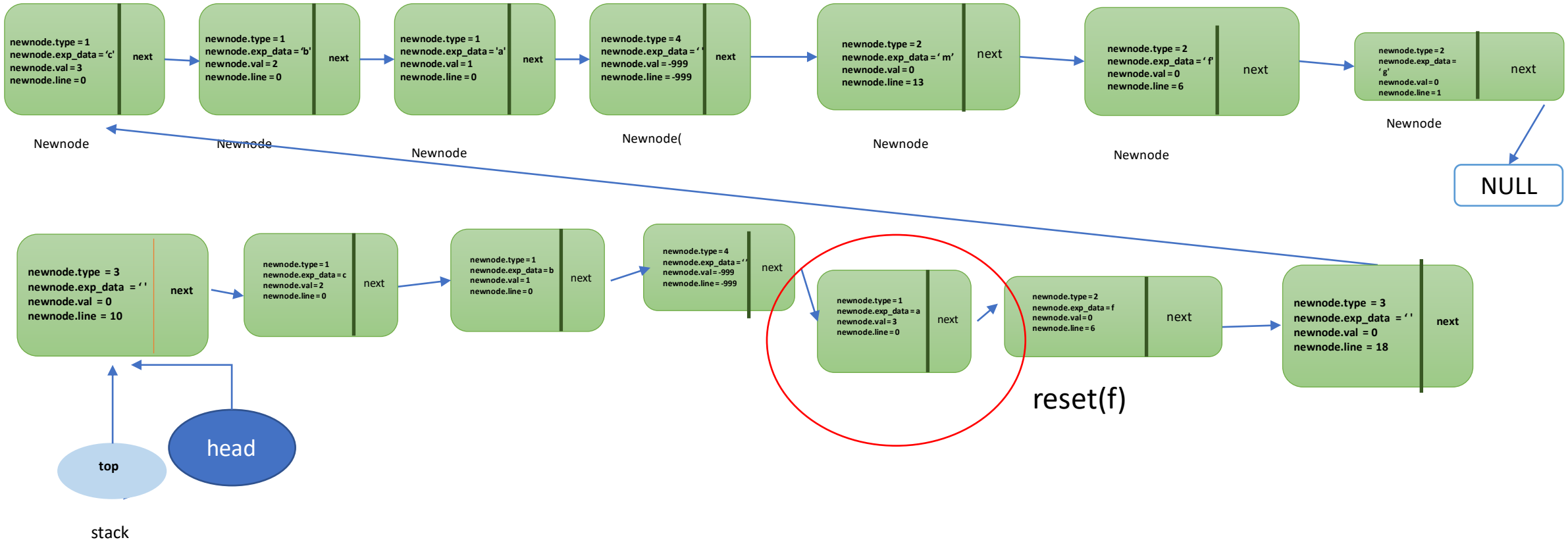
1

push()


newnode.type = 1
newnode.exp_data = 'c'
newnode.val = 3
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'b'
newnode.val = 2
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'a'
newnode.val = 1
newnode.line = 0
next
Newnode
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
Newnode(
newnode.type = 2
newnode.exp_data = ' m'
newnode.val = 0
newnode.line = 13
next
Newnode
newnode.type = 2
newnode.exp_data = ' f'
newnode.val = 0
newnode.line = 6
next
Newnode
newnode.type = 2
newnode.exp_data = ' g'
newnode.val = 0
newnode.line = 1
next
Newnode
NULL
newnode.type = 3
newnode.exp_data = ' '
newnode.val = 0
newnode.line = 10
next
top
stack
newnode.type = 1
newnode.exp_data = c
newnode.val = 2
newnode.line = 0
next
newnode.type = 1
newnode.exp_data = b
newnode.val = 1
newnode.line = 0
next
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
newnode.type = 1
newnode.exp_data = a
newnode.val = 3
newnode.line = 0
next
newnode.type = 2
newnode.exp_data = f
newnode.val = 0
newnode.line = 6
next
reset(f)
newnode.type = 3
newnode.exp_data = ' '
newnode.val = 0
newnode.line = 18
next

2

getVal()


newnode.type = 1
newnode.exp_data = 'c'
newnode.val = 3
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'b'
newnode.val = 2
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'a'
newnode.val = 1
newnode.line = 0
next
Newnode
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
Newnode(
newnode.type = 2
newnode.exp_data = ' m'
newnode.val = 0
newnode.line = 13
next
Newnode
newnode.type = 2
newnode.exp_data = ' f'
newnode.val = 0
newnode.line = 6
next
Newnode
newnode.type = 2
newnode.exp_data = ' g'
newnode.val = 0
newnode.line = 1
next
Newnode
NULL
newnode.type = 3
newnode.exp_data = ' '
newnode.val = 0
newnode.line = 10
next
newnode.type = 1
newnode.exp_data = c
newnode.val = 2
newnode.line = 0
next
newnode.type = 1
newnode.exp_data = b
newnode.val = 1
newnode.line = 0
next
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
newnode.type = 1
newnode.exp_data = a
newnode.val = 3
newnode.line = 0
next
newnode.type = 2
newnode.exp_data = f
newnode.val = 0
newnode.line = 6
next
newnode.type = 3
newnode.exp_data = ' '
newnode.val = 0
newnode.line = 18
next
reset(f)
top
head
stack

```
    if (LastFunctionReturn==-999)
    {
    /* if function */
    /* add to our system stack that we are making a call to function */
    int j;
    tempNode.type=3;
    tempNode.line=curLine;
    STACK=Push(tempNode,STACK);

    /* get function's arguments value */
    CalingFunctionArgVal=GetVal(lineyedek[i+2],&dummyint,STACK);



    fclose(filePtr);
    filePtr=fopen(argv[1],"r");
    curLine=0;
    /* file reversed to start postion */
    /* now go codeline lines to go, to the functions line */


codeline olabilir */
    for(j=1;j<codeline;j++)
    {
        fgets(dummy,4096,filePtr); /* read the file by Line by Line */
        curLine++;
    }

    WillBreak=1;
    break;
    }
```

break를 만나 whil문을
빠져나간다.

```
//lineyedek[i]이 NULL이 아닐때까지 반복
while(lineyedek[i]!='₩x0')
{
    //(3)-3-1
    /* evulate the function */
    //숫자라면 true
    if (isdigit(lineyedek[i])) {
        postfix[y]=lineyedek[i];

        y++;  //1증가

    }
     /* ... */

    //(3)-3-2
    //lineyedek[i]이 ')'이라면
    else if (lineyedek[i]==')')
    {
        //(3)-3-2-1
        ////0이면 true 1이면 false
        //0일때
        if (!isStackEmpty(MathStack) != 0 ) //- OpStack->top==0이 아닐 때
        {
            postfix[y]=PopOp(MathStack);
            y++; // 1증가
        }
    }

    ////(3)-3-3
    //lineyedek[i]이 '+'이거나 lineyedek[i]이 '-' 이거나 lineyedek[i]이 '*' 이거나 lineyedek[i]이 '/'이라면
    else if ((lineyedek[i]=='+') | (lineyedek[i]=='-') | (lineyedek[i]=='*') | (lineyedek[i]=='/')){ ... }

    ////(3)-3-4
    //알파벳 대문자 'A-Z'는 1을 반환.알파벳 소문자 'a-z'는 2를 반환.
    //lineyedek[i] 영어라면
    else if (isalpha(lineyedek[i])>0){ ... }

    //1 증가
    i++;
}//while문



//WillBreak이 0이라면
if (WillBreak==0){ ... }
WillBreak=0;
}
```

다시 reset

function g(int x)

```
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //fget을 통해 function f(int a)을 읽어 line에 저장 .
    fgets(line,4096,filePtr); /* read the file by Line by Line */
    /* scan for /t characters, get rid of them! */

    // whil문 line[k]이  null 이 아닐 시때까지 반복.
    while(line[k]!='₩0')
    {
       //  4-1) 만약 line[k]가 ₩t(탭키)일 시,
     if (line[k]=='₩t')
     {
       //line[k]는  ' '으로 바꾼다.
      line[k]=' ';
     }
      //k에 1증가
     k++;
    }
    //5. line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을  ' '로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //begin과 line이 동일하다면
    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }


    //ⓑ
    //  //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //line이 end라면
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }

    //ⓒ begin과 end가 아니면
    else{ ... }
```

input 2.sql (읽을 파일)

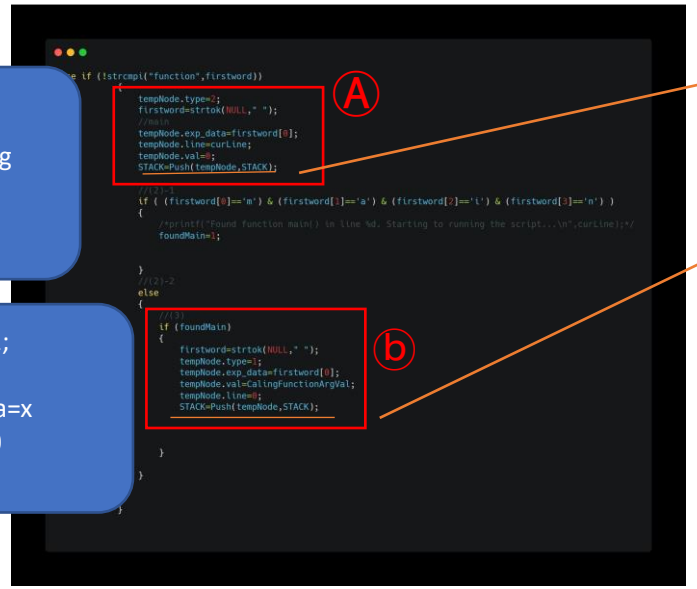
```
function g(int x)
begin
  (1+2-3+x);
end

function f(int a)
begin
  int b = 1;
  int c = 2;
  ((b*c)+g(a));
end

function main()
begin
  int a = 1;
  int b = 2;
  int c = 3;
  ((2 + f(c) ) * a);
end
```

다시 reset

function g(int x)

tempNode.type=2;
firstword= g(int
tempNode.exp_data=g
tempNode.line=1
tempNode.val=0;

tempNode.type=1;
firstword= x)
tempNode.exp_data=x
tempNode.line=0
tempNode.val=3



파일을 다시 새로 시작한다.

Push

```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```

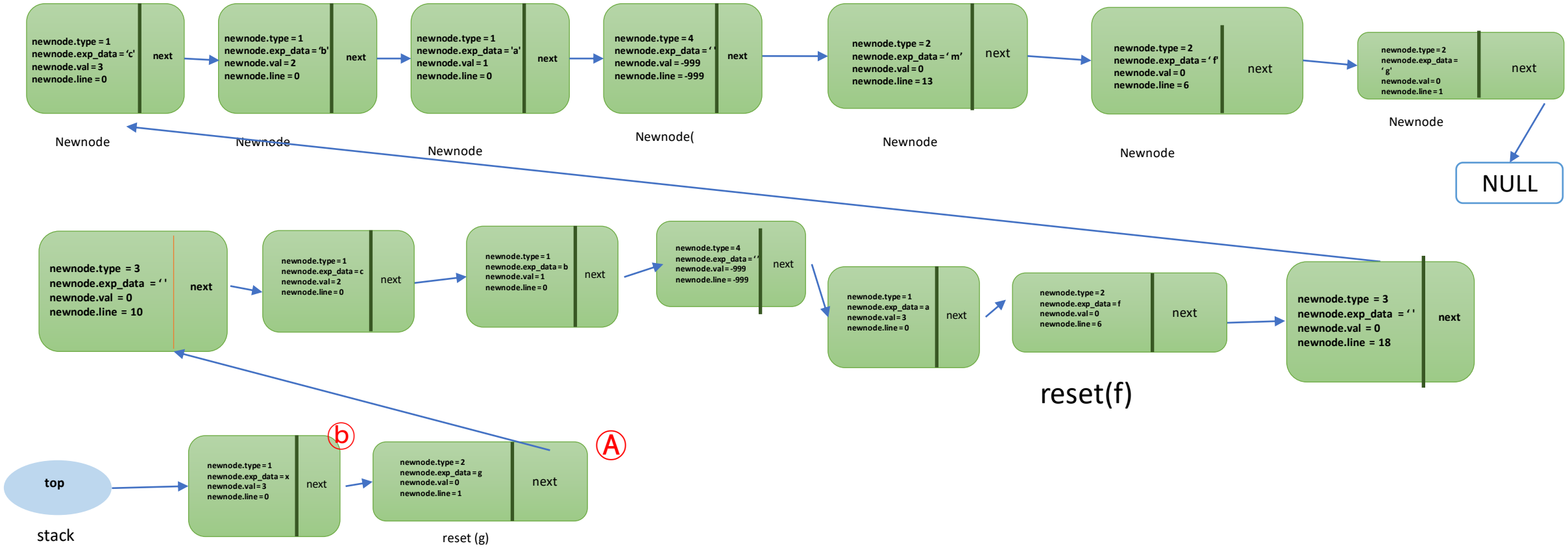
Stack * Push(Node sNode,Stack *stck)

1. Node 자료형의 newnode포인터선언;
2. 만약 newnode는 Node타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   return NULL

3.아니라면
   newnode의 type은 sNode의 타입이 된다.
   newnode 의 val은 sNode의 val이된다.
   newnode 의 exp_data은 sNode의 exp_data이된다.
   newnode 의 line 은 sNode의 line이 된다.
   newnode 의 next 는 stck의 top이 된다.
   stck의 top은 newnode이다

반환값 stck

function


newnode.type = 1
newnode.exp_data = 'c'
newnode.val = 3
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'b'
newnode.val = 2
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'a'
newnode.val = 1
newnode.line = 0
next
Newnode
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
Newnode(
newnode.type = 2
newnode.exp_data = ' m'
newnode.val = 0
newnode.line = 13
next
Newnode
newnode.type = 2
newnode.exp_data = ' f'
newnode.val = 0
newnode.line = 6
next
Newnode
newnode.type = 2
newnode.exp_data = ' g'
newnode.val = 0
newnode.line = 1
next
Newnode
NULL
newnode.type = 3
newnode.exp_data = ' '
newnode.val = 0
newnode.line = 10
next
newnode.type = 1
newnode.exp_data = c
newnode.val = 2
newnode.line = 0
next
newnode.type = 1
newnode.exp_data = b
newnode.val = 1
newnode.line = 0
next
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
newnode.type = 1
newnode.exp_data = a
newnode.val = 3
newnode.line = 0
next
newnode.type = 2
newnode.exp_data = f
newnode.val = 0
newnode.line = 6
next
newnode.type = 3
newnode.exp_data = ' '
newnode.val = 0
newnode.line = 18
next
reset(f)
top
stack
newnode.type = 1
newnode.exp_data = x
newnode.val = 3
newnode.line = 0
next
b
newnode.type = 2
newnode.exp_data = g
newnode.val = 0
newnode.line = 1
next
A
reset (g)

다시 reset

begin

```
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //fget을 통해 function f(int a)을 읽어 line에 저장 .
    fgets(line,4096,filePtr); /* read the file by Line by Line */
    /* scan for /t characters, get rid of them! */

    // whil문 line[k]이  null 이 아닐 시때까지 반복.
    while(line[k]!='₩0')
    {
       //  4-1) 만약 line[k]가 ₩t(탭키)일 시,
     if (line[k]=='₩t')
     {
       //line[k]는  ' '으로 바꾼다.
      line[k]=' ';
     }
      //k에 1증가
     k++;
    }
    //5. line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을  ' '로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //begin과 line이 동일하다면
    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }


    //ⓑ
    //  //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //line이 end라면
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }

    //ⓒ begin과 end가 아니면
    else{ ... }
```

input 2.sql (읽을 파일)

```
function g(int x)
begin
  (1+2-3+x);
end

function f(int a)
begin
   int b = 1;
   int c = 2;
   ((b*c)+g(a));
end

function main()
begin
   int a = 1;
   int b = 2;
   int c = 3;
   ((2 + f(c) ) * a);
end
```

begin

```
// ⓐ하나
//strcmpi :  둘이 동일하면 1(true) 반환하고 동일하지 않으면 0(false)
//begin과 line이 동일하다면
if (!strcmpi("beginMn",line) | !strcmpi("begin",line))
{
    //foundMain == 0이면 false 0이 아니면 True
    if (foundMain)
    {
        //tempNode.type 는 4이다.
        tempNode.type=4;
        ////STACK은 Push(tempNode,STACK); 에서의 반환값이다.
        STACK=Push(tempNode,STACK);
    }
}
```

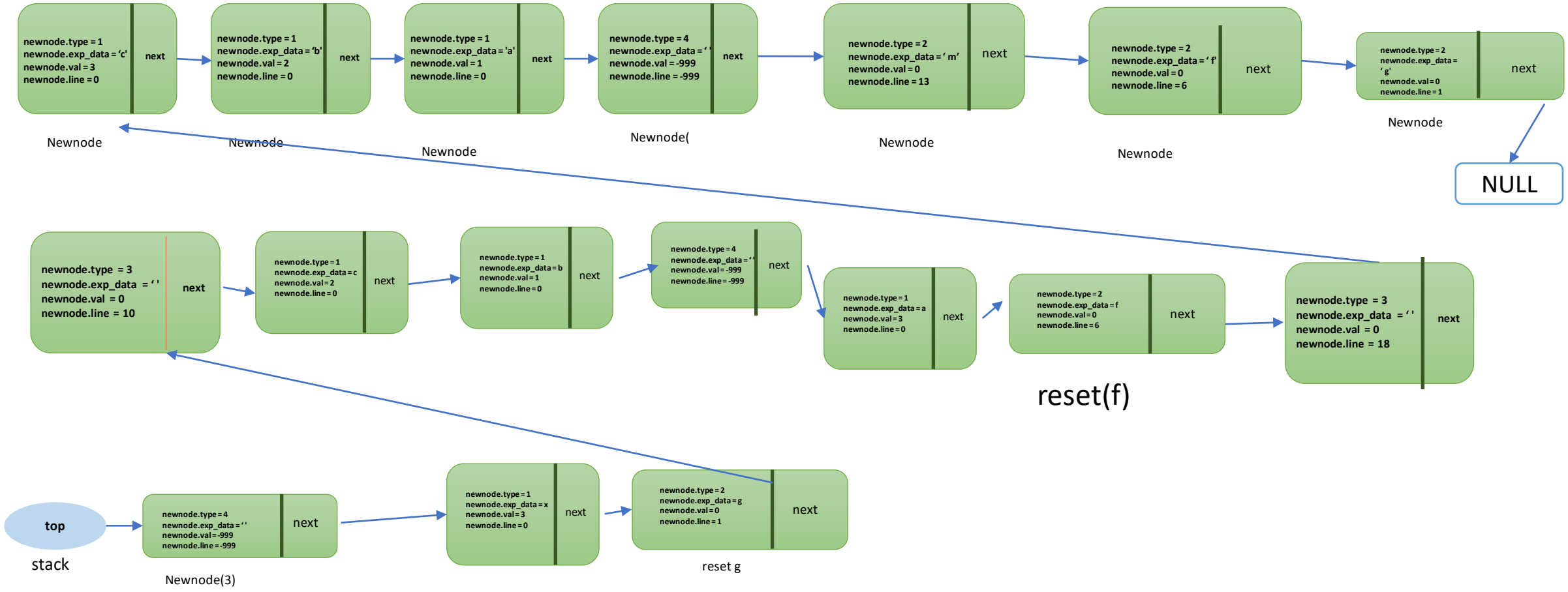
Push

```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```

Node자료형의newnode포인터선언;
만약 newnode는 Node타입크기 만큼메모리를 할당받으며,
newnode가NULL일 경우 ERROR, Couldn't allocate memory... 출력
NULL리턴

newnode.type = 4
newnode.val = -999
newnode.exp_data = ' '
newnode.line = -999
newnode의 next 는 stck의 top이 된다.
stck의 top은 newnode이다
반환값 stck

Push() 후


newnode.type = 1
newnode.exp_data = 'c'
newnode.val = 3
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'b'
newnode.val = 2
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'a'
newnode.val = 1
newnode.line = 0
next
Newnode
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
Newnode(
newnode.type = 2
newnode.exp_data = ' m'
newnode.val = 0
newnode.line = 13
next
Newnode
newnode.type = 2
newnode.exp_data = ' f'
newnode.val = 0
newnode.line = 6
next
Newnode
newnode.type = 2
newnode.exp_data = ' g'
newnode.val = 0
newnode.line = 1
next
Newnode
NULL
newnode.type = 3
newnode.exp_data = ' '
newnode.val = 0
newnode.line = 10
next
newnode.type = 1
newnode.exp_data = c
newnode.val = 2
newnode.line = 0
next
newnode.type = 1
newnode.exp_data = b
newnode.val = 1
newnode.line = 0
next
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
newnode.type = 1
newnode.exp_data = a
newnode.val = 3
newnode.line = 0
next
newnode.type = 2
newnode.exp_data = f
newnode.val = 0
newnode.line = 6
next
newnode.type = 3
newnode.exp_data = ' '
newnode.val = 0
newnode.line = 18
next
reset(f)
top
stack
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
Newnode(3)
newnode.type = 1
newnode.exp_data = x
newnode.val = 3
newnode.line = 0
next
newnode.type = 2
newnode.exp_data = g
newnode.val = 0
newnode.line = 1
next
reset g

다시 reset

begin

```
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //fget을 통해 function f(int a)을 읽어 line에 저장 .
    fgets(line,4096,filePtr); /* read the file by Line by Line */
    /* scan for /t characters, get rid of them! */

    // whil문 line[k]이  null 이 아닐 시때까지 반복.
    while(line[k]!='₩0')
    {
       //  4-1) 만약 line[k]가 ₩t(탭키)일 시,
     if (line[k]=='₩t')
     {
       //line[k]는  ' '으로 바꾼다.
      line[k]=' ';
     }
      //k에 1증가
     k++;
    }
    //5. line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을  ' '로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //begin과 line이 동일하다면
    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }


    //ⓑ
    //  //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //line이 end라면
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }

    //ⓒ begin과 end가 아니면
    else{ ... }
```

input 2.sql (읽을 파일)

```
function g(int x)
begin
 (1+2-3+x);
end

function f(int a)
begin
  int b = 1;
  int c = 2;
  ((b*c)+g(a));
end

function main()
begin
  int a = 1;
  int b = 2;
  int c = 3;
  ((2 + f(c) ) * a);
end
```

(1+2-3+x);

firstword[0]= (라면

```
else if (firstword[0]=='(')
{

//foundMain == 0이면 false 0이 아니면 True
if (foundMain)
    {

    int i=0;
    int y=0; //3

    //OpStack + MathStack
    //MathStack.top은 NULL
    MathStack->top=NULL;
    /* now make the postfix calculcation */



    //(3)-2
    //lineyedek[i]이 NULL이 아닐때까지 반복
    while(lineyedek[i]!='₩x0')
    {
        //(3)-3-1
        /* evulate the function */
        //숫자라면 true
        if (isdigit(lineyedek[i])){ ... }
         /* ... */

        //(3)-3-2
        //lineyedek[i]이 ')'이라면
        else if (lineyedek[i]==')'){ ... }

        ////(3)-3-3
        //lineyedek[i]이 '+'이거나 lineyedek[i]이 '-' 이거나 lineyedek[i]이 '+' 이거나 lineyedek[i]이 '/'이라면
        else if ((lineyedek[i]=='+') | (lineyedek[i]=='-') | (lineyedek[i]=='+') | (lineyedek[i]=='/')){ ... }

        ////(3)-3-4
        //알파벳 대문자 "A-Z"는 1를 반환,알파벳 소문자 'a-z"는 2를 반환.
        //lineyedek[i] 영어라면
        else if (isalpha(lineyedek[i])>0){ ... }

        //i 증가
        i++;
    }//while문
```

```
/* evulate the function */
//숫자라면 true
if (isdigit(lineyedek[i])) {
    postfix[y]=lineyedek[i];
    y++;

}
 /*      */
```

postfix[0] = 1

(1+2-3+x);

만약 isStackEmpty( ) 0이아니라면

```
////(3)-3-3
//lineyedek[i]이 '+'이거나 lineyedek[i]이 '-' 이거나 lineyedek[i]이 '+' 이거나 lineyedek[i]이 '/'이라면
else if ((lineyedek[i]=='+') | (lineyedek[i]=='-') | (lineyedek[i]=='*') | (lineyedek[i]=='/'))
{
   ////(3)-3-3-1
   /*operators*/
   //OO이면 false 1이면 true
   //처음일 때
   if (isStackEmpty(MathStack) != 0 ) //1일때 - stck->top==0
   {
/* if stack empty push the operator to stack */
      //+
      MathStack=PushOp(lineyedek[i],MathStack);
   }
   ////(3)-3-3-2

   //0일때 - stck -top = 00| 아닐때
   //처음이 아닐때
   else { ... }
}
```

ⓐ

ⓑ

PushOp(+, MathStack)

isStackEmpty

```
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)

1. 만약 stck의 top가 0이라면
   1. 반환값 1
2. 아니면 반환값 0

PushOp

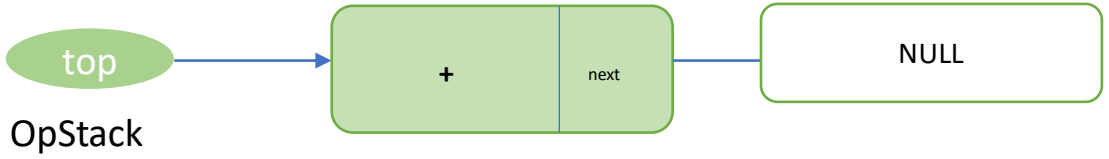
```
OpStack * PushOp(char op,OpStack *opstck)
{
opNode *newnode;
if ((newnode=(opNode*)malloc(sizeof(opNode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->op=op;
newnode->next=opstck->top;
opstck->top=newnode;
return opstck;
}
}
```

OpStack * PushOp(char op,OpStack *opstck)

1. opNode자료형의 newnode포인터선언;
2. 만약 newnode는 opNode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   반환NULL
3. 아니라면,
   1. newnode의 op는 매개변수+이다.
   2. newnode의 next는 opstck의 top이다.
   3. opstck의 top은 newnode이다.

반환값 opstck

PushOp



(1+2-3+x);

firstword[0]= (라면

```
else if (firstword[0]=='(')
{

//foundMain == 0이면 false 0이 아니면 True
if (foundMain)
    {

    int i=0;
    int y=0; //3

    //OpStack + MathStack
    //MathStack.top은 NULL
    MathStack->top=NULL;
    /* now make the postfix calculcation */



    //(3)-2
    //lineyedek[i]이 NULL이 아닐때까지 반복
    while(lineyedek[i]!='₩x0')
    {
        //(3)-3-1
        /* evulate the function */
        //숫자라면 true
        if (isdigit(lineyedek[i])){ ... }
         /* ... */

        //(3)-3-2
        //lineyedek[i]이 ')'이라면
        else if (lineyedek[i]==')'){ ... }

        ////(3)-3-3
        //lineyedek[i]이 '+'이거나 lineyedek[i]이 '-' 이거나 lineyedek[i]이 '+' 이거나 lineyedek[i]이 '/'이라면
        else if ((lineyedek[i]=='+') | (lineyedek[i]=='-') | (lineyedek[i]=='+') | (lineyedek[i]=='/')){ ... }

        ////(3)-3-4
        //알파벳 대문자 "A-Z"는 1를 반환,알파벳 소문자 'a-z"는 2를 반환.
        //lineyedek[i] 영어라면
        else if (isalpha(lineyedek[i])>0){ ... }

        //1 증가
        i++;
    }//while문
```

```
/* evulate the function */
//숫자라면 true
if (isdigit(lineyedek[i])) {
    postfix[y]=lineyedek[i];
    y++;

}
 /*      */
```

postfix[1] = 2

(1+2-3+x);

만약 isStackEmpty( ) 0이아니라면

```
////(3)-3-3
//lineyedek[i]이 '+'이거나 lineyedek[i]이 '-' 이거나 lineyedek[i]이 '+' 이거나 lineyedek[i]이 '/'이라면
else if ((lineyedek[i]=='+') | (lineyedek[i]=='-') | (lineyedek[i]=='*') | (lineyedek[i]=='/'))
{
   ////(3)-3-3-1
   /*operators*/
   //00이면 false 10이면 true
   //처음일 때
   if (isStackEmpty(MathStack) != 0 ) //1일때 - stck->top==0
   {
/* if stack empty push the operator to stack */
      //+
      MathStack=PushOp(lineyedek[i],MathStack);
   }
   ////(3)-3-3-2

   //0일때 - stck -top = 00이 아닐때
   //처음이 아닐때
   else { ... }
}
```

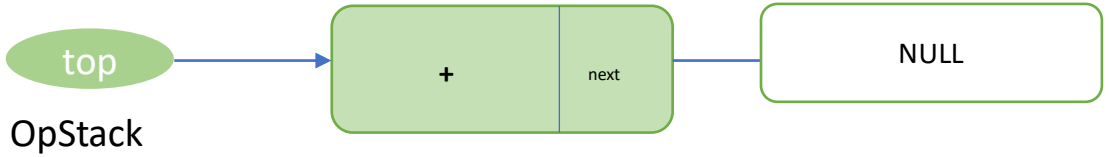
isStackEmpty

```
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)

1. 만약 stck의 top가 0이라면
   1. 반환값 1
2. 아니면 반환값 0

PushOp



(1+2-3+x);

```
//isStackEmpty(MathStack)의 반환값이 0이라면
else
{
    //Priotry(MathStack->top->op)의 반환값이 Priotry(lineyedek[i])의 반환값보다 크거나 같을때
    if (Priotry(lineyedek[i]) <= Priotry(MathStack->top->op) )
    {
        //postfix[y]은 PopOp(MathStack)반환값
        postfix[y]=PopOp(MathStack);

        //y값 1증가
        y++;

        //MathStack은 PushOp(lineyedek[i],MathStack)이다
        MathStack=PushOp(lineyedek[i],MathStack);


    }
    // Priotry(lineyedek[i])의 반환값이 Priotry(MathStack->top->op)의 반환값보다 클 때
    else
    {
        //MathStack은 PushOp(lineyedek[i],MathStack)의 반환값
        MathStack=PushOp(lineyedek[i],MathStack);
    }

}
}
```

Priotry

```
int Priotry(char operator)
{
  if ((operator=='+') | (operator=='-'))
    return 1;
  else if ((operator=='/') | (operator=='*'))
    return 2;
return 0;
}
```

int Priotry(char operator)

1. 만약 operator 가 '+'이거나 operator가 '-'라면
    1. 반환값 1
2. 만약 operator가 '/'이거나 operator가 '*'이라면
    1. 반환값 2
3. 반환값 0

(Priotry(lineyedek[i]) = 1 이며,

Priotry(MathStack->top->op) =1

1

Priotry(MathStack->top->op)의 반환값이
Priotry(lineyedek[i])의 반환값보다 크거나 같을때

```
//*)*(*
//isStackEmpty(MathStack)이 OO이라면
else
{
    //Priotry(MathStack->top->op이 Priotry(lineyedek[i])의 반환값보다 크거나 같을때
    if (Priotry(lineyedek[i]) <= Priotry(MathStack->top->op) )
    {
        //postfix[y]은 PopOp(MathStack)반환값
        postfix[y]=PopOp(MathStack);
        //y값 1증가
        y++;

        //MathStack은 PushOp(lineyedek[i],MathStack)이다
        MathStack=PushOp(lineyedek[i],MathStack);


    }
    //Priotry(MathStack->top->op이 Priotry(lineyedek[i])이 작을 때
    else
    {
        //MathStack은 PushOp(lineyedek[i],MathStack)의 반환값
        MathStack=PushOp(lineyedek[i],MathStack);
    }

}
```

ⓐ

PopOp

```
char PopOp(OpStack *opstck)
{
opNode *temp;
char op;
if (opstck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
op=opstck->top->op;
temp=opstck->top;
opstck->top=opstck->top->next;
free(temp);
return op;
}
return NULL;
}
```

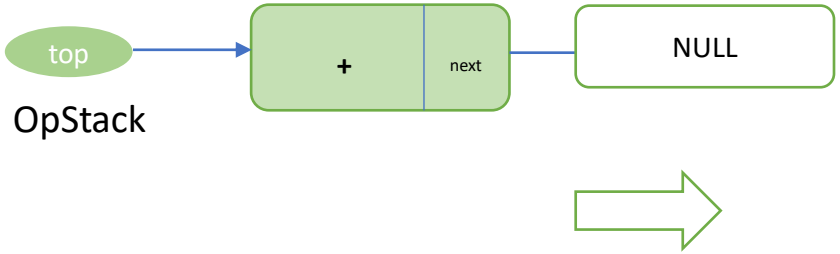
char PopOp(OpStack *opstck)

1. opNode구조체의 자료형 temp 포인터 선언
2. 문자자료형 op선언
3. 만약 opstck의 top이 NULL일 경우
   printf("ERROR, empty stack..."); 출력
4. 아니라면,
   1. op는 opstck의 top의 op이다
   2. temp 는 opstck의 top이다.
   3. opstck의 top은 opstck의 top의 next이다.
   temp 동적메모리 해제
   반환값 op

반환값 NULL

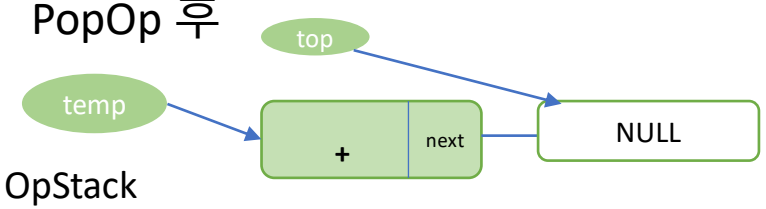
예시)

전



postfix[2] = +

PopOp 후



1

Priotry(MathStack->top->op)의 반환값이
Priotry(lineyedek[i])의 반환값보다 크거나 같을때

```
//*)*(*
//isStackEmpty(MathStack)이 OO이라면
else
{
    //Priotry(MathStack->top->op이 Priotry(lineyedek[i])의 반환값보다 크거나 같을때
    if (Priotry(lineyedek[i]) <= Priotry(MathStack->top->op) )
    {
        //postfix[y]은 PopOp(MathStack)반환값
        postfix[y]=PopOp(MathStack);
        //y값 1증가
        y++;

        //MathStack은 PushOp(lineyedek[i],MathStack)이다
        MathStack=PushOp(lineyedek[i],MathStack);

    }
    //Priotry(MathStack->top->op이 Priotry(lineyedek[i])이 작을 때
    else
    {
        //MathStack은 PushOp(lineyedek[i],MathStack)의 반환값
        MathStack=PushOp(lineyedek[i],MathStack);
    }

}
```

PushOp

```
OpStack * PushOp(char op,OpStack *opstck)
{
opNode *newnode;
if ((newnode=(opNode*)malloc(sizeof(opNode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->op=op;
newnode->next=opstck->top;
opstck->top=newnode;
return opstck;
}
}
```

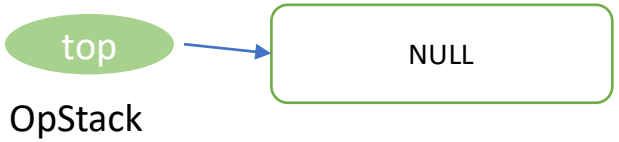
OpStack * PushOp(char op,OpStack *opstck)

1. opNode자료형의 newnode포인터선언;
2. 만약 newnode는 opNode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   반환NULL
3. 아니라면,
   1. newnode의 op는 매개변수+이다.
   2. newnode의 next는 opstck의 top이다.
   3. opstck의 top은 newnode이다.

반환값 opstck

예시)

PushOp 전



PushOp 후

(1+2-3+x);

firstword[0]= (라면

```
else if (firstword[0]=='(')
{

//foundMain == 0이면 false 0이 아니면 True
if (foundMain)
   {

   int i=0;
   int y=0; //3

   //OpStack + MathStack
   //MathStack.top은 NULL
   MathStack->top=NULL;
   /* now make the postfix calculcation */



   //(3)-2
   //lineyedek[i]이 NULL이 아닐때까지 반복
   while(lineyedek[i]!='₩x0')
   {
      //(3)-3-1
      /* evulate the function */
      //숫자라면 true
      if (isdigit(lineyedek[i])){ ... }
       /* ... */

      //(3)-3-2
      //lineyedek[i]이 ')'이라면
      else if (lineyedek[i]==')'){ ... }

      ////(3)-3-3
      //lineyedek[i]이 '+'이거나 lineyedek[i]이 '-' 이거나 lineyedek[i]이 '+' 이거나 lineyedek[i]이 '/'이라면
      else if ((lineyedek[i]=='+') | (lineyedek[i]=='-') | (lineyedek[i]=='+') | (lineyedek[i]=='/')){ ... }

      ////(3)-3-4
      //알파벳 대문자 "A-Z"는 1를 반환,알파벳 소문자 'a-z"는 2를 반환.
      //lineyedek[i] 영어라면
      else if (isalpha(lineyedek[i])>0){ ... }

      //i 증가
      i++;
   }//while문
```

```
/* evulate the function */
//숫자라면 true
if (isdigit(lineyedek[i])) {
   postfix[y]=lineyedek[i];
   y++;

}
 /*      */
```

postfix[3] = 3

(1+2-3+x);

만약 isStackEmpty( ) 0이아니라면

```
////(3)-3-3
//lineyedek[i]이 '+'이거나 lineyedek[i]이 '-' 이거나 lineyedek[i]이 '+' 이거나 lineyedek[i]이 '/'이라면
else if ((lineyedek[i]=='+') || (lineyedek[i]=='-') || (lineyedek[i]=='*') || (lineyedek[i]=='/'))
{
   ////(3)-3-3-1
   /*operators*/
   //00이면 false 10이면 true
   //처음일 때
   if (isStackEmpty(MathStack) != 0 ) //1일때 - stck->top==0
   {
/* if stack empty push the operator to stack */
      //+
      MathStack=PushOp(lineyedek[i],MathStack);
   }
   ////(3)-3-3-2

   //0일때 - stck -top = 00이 아닐때
   //처음이 아닐때
   else { ... }
}
```

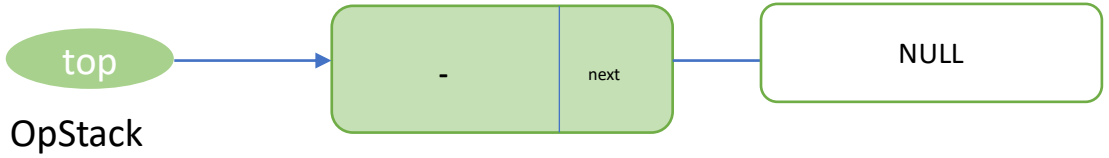
isStackEmpty

```
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)

1. 만약 stck의 top가 0이라면
   1. 반환값 1
2. 아니면 반환값 0

PushOp



(1+2-3+x);

```
//isStackEmpty(MathStack)의 반환값이 0이라면
else
{
    //Priotry(MathStack->top->op)의 반환값이 Priotry(lineyedek[i])의 반환값보다 크거나 같을때
    if (Priotry(lineyedek[i]) <= Priotry(MathStack->top->op) )
    {
        //postfix[y]은 PopOp(MathStack)반환값
        postfix[y]=PopOp(MathStack);

        //y값 1증가
        y++;

        //MathStack은 PushOp(lineyedek[i],MathStack)이다
        MathStack=PushOp(lineyedek[i],MathStack);


    }
    // Priotry(lineyedek[i])의 반환값이 Priotry(MathStack->top->op)의 반환값보다 클 때
    else
    {
        //MathStack은 PushOp(lineyedek[i],MathStack)의 반환값
        MathStack=PushOp(lineyedek[i],MathStack);
    }

}
}
```

Priotry

```
int Priotry(char operator)
{
  if ((operator=='+') | (operator=='-'))
    return 1;
  else if ((operator=='/') | (operator=='*'))
    return 2;
return 0;
}
```

int Priotry(char operator)

1. 만약 operator 가 '+'이거나 operator가 '-'라면
    1. 반환값 1
2. 만약 operator가 '/'이거나 operator가 '*'이라면
    1. 반환값 2
3. 반환값 0

(Priotry(lineyedek[i]) = 1 이며,

Priotry(MathStack->top->op) =1

1

Priotry(MathStack->top->op)의 반환값이
Priotry(lineyedek[i])의 반환값보다 크거나 같을때

```
//*)*(*
//isStackEmpty(MathStack)이 OO이라면
else
{
    //Priotry(MathStack->top->op이 Priotry(lineyedek[i])의 반환값보다 크거나 같을때
    if (Priotry(lineyedek[i]) <= Priotry(MathStack->top->op) )
    {
        //postfix[y]은 PopOp(MathStack)반환값
        postfix[y]=PopOp(MathStack);
        //y값 1증가
        y++;

        //MathStack은 PushOp(lineyedek[i],MathStack)이다
        MathStack=PushOp(lineyedek[i],MathStack);


    }
    //Priotry(MathStack->top->op이 Priotry(lineyedek[i])이 작을 때
    else
    {
        //MathStack은 PushOp(lineyedek[i],MathStack)의 반환값
        MathStack=PushOp(lineyedek[i],MathStack);
    }

}
```

ⓐ

PopOp

```
char PopOp(OpStack *opstck)
{
opNode *temp;
char op;
if (opstck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
op=opstck->top->op;
temp=opstck->top;
opstck->top=opstck->top->next;
free(temp);
return op;
}
return NULL;
}
```

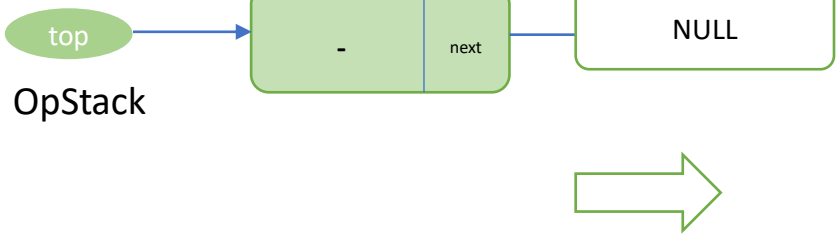
char PopOp(OpStack *opstck)

1. opNode구조체의 자료형 temp 포인터 선언
2. 문자자료형 op선언
3. 만약 opstck의 top이 NULL일 경우
   printf("ERROR, empty stack..."); 출력
4. 아니라면,
   1. op는 opstck의 top의 op이다
   2. temp 는 opstck의 top이다.
   3. opstck의 top은 opstck의 top의 next이다.
   temp 동적메모리 해제
   반환값 op

반환값 NULL

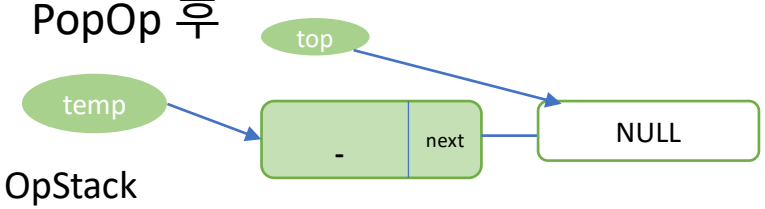
예시)

전



postfix[4] = -

PopOp 후



1

Priotry(MathStack->top->op)의 반환값이
Priotry(lineyedek[i])의 반환값보다 크거나 같을때

```
//*)*(*
//isStackEmpty(MathStack)이 아니라면
else
{
    //Priotry(MathStack->top->op이 Priotry(lineyedek[i])의 반환값보다 크거나 같을때
    if (Priotry(lineyedek[i]) <= Priotry(MathStack->top->op) )
    {
        //postfix[y]은 PopOp(MathStack)반환값
        postfix[y]=PopOp(MathStack);
        //y값 1증가
        y++;

        //MathStack은 PushOp(lineyedek[i],MathStack)이다
        MathStack=PushOp(lineyedek[i],MathStack);

    }
    //Priotry(MathStack->top->op이 Priotry(lineyedek[i])이 작을 때
    else
    {
        //MathStack은 PushOp(lineyedek[i],MathStack)의 반환값
        MathStack=PushOp(lineyedek[i],MathStack);
    }

}
```

PushOp

```
OpStack * PushOp(char op,OpStack *opstck)
{
opNode *newnode;
if ((newnode=(opNode*)malloc(sizeof(opNode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->op=op;
newnode->next=opstck->top;
opstck->top=newnode;
return opstck;
}
}
```

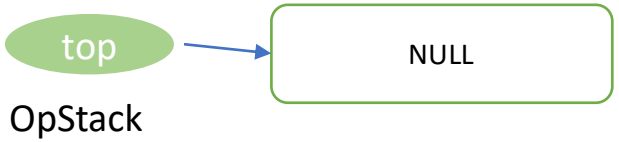
OpStack * PushOp(char op,OpStack *opstck)

1. opNode자료형의 newnode포인터선언;
2. 만약 newnode는 opNode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   반환NULL
3. 아니라면,
   1. newnode의 op는 매개변수+이다.
   2. newnode의 next는 opstck의 top이다.
   3. opstck의 top은 newnode이다.

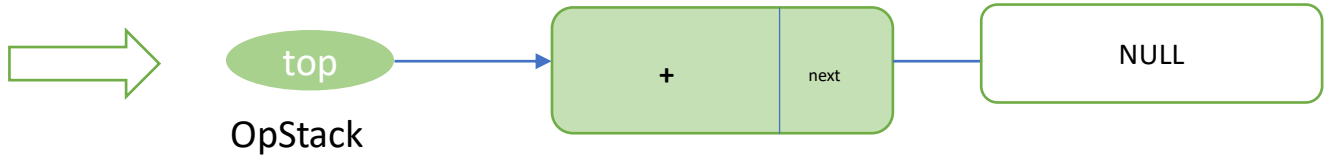
반환값 opstck

예시)

PushOp 전



PushOp 후



(1+2-3+x);

```
//lineyedek[i] 영어라면
else if (isalpha(lineyedek[i])>0)
{
    int codeline=0;
    int dummyint=0;
    /*look if it's a variable or function call
    int retVal=0;
    retVal=GetVal(lineyedek[i],&codeline,STACK);
    //int codeline =1

    //-1


    ////(3)-3-4-1
    if ((retVal!=-1) & (retVal!=-999)){ ... }
    ////(3)-3-4-2
    //-1
    else{ ... }
}
```

GetVal (x,&codeline,STACK)

GetVal

```
int GetVal(char exp_name,int * line,Stack *stck)
{

Node * head;
*line=0;
    if (stck->top == NULL )
    {
        printf("ERROR, empty stack...");
    }
    else
    {
        head=stck->top;
        do
        {
            if ( head->exp_data==exp_name )
            {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }

            }
            else
            {
                head=head->next;
            }
        } while (head->next!=NULL);
        /* check agin once more */
        if (head->exp_data==exp_name)
        {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }


        }
    }

    return -999;
}
```

head->exp_data = x

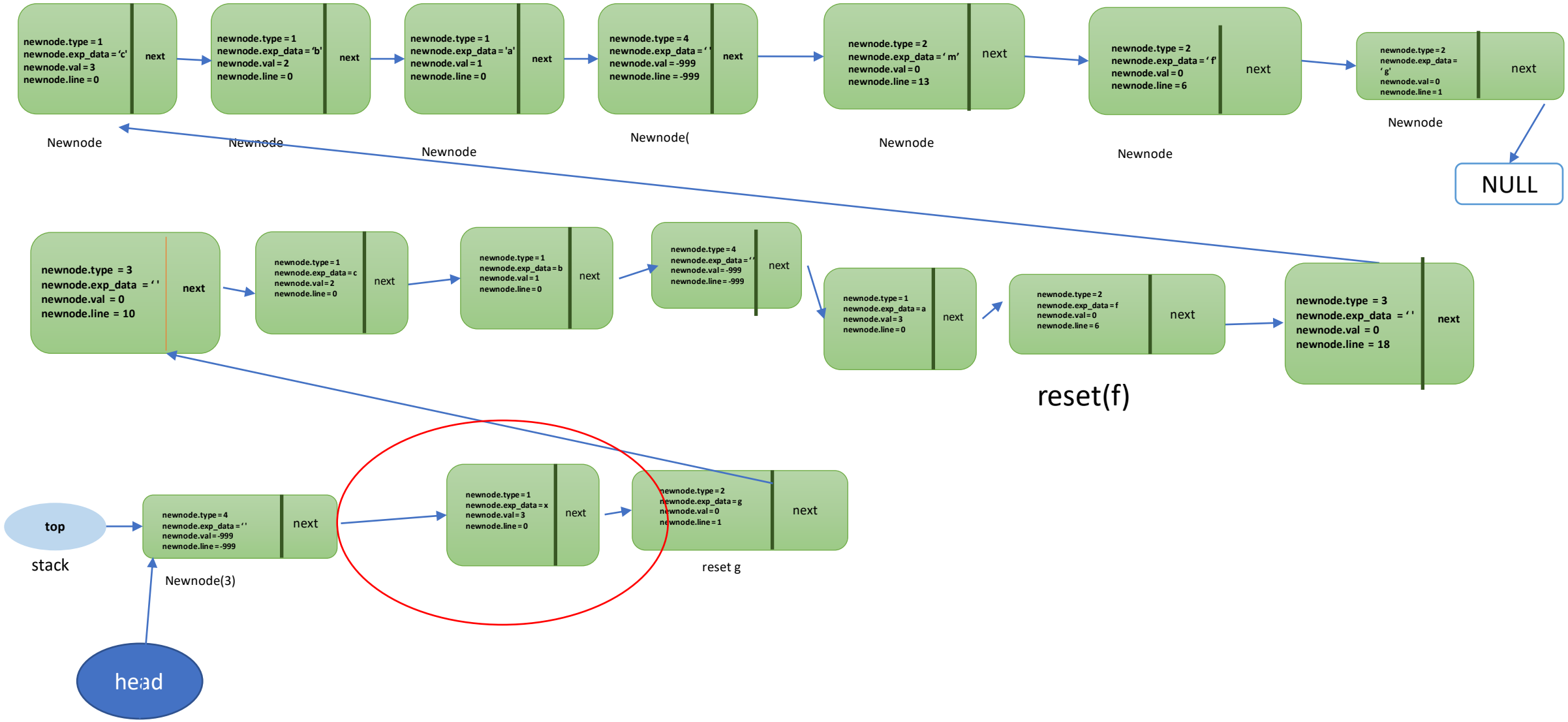
return 1

GetVal()
자료형인 Node인 포인트 head선언

1. head를 stck의 top으로 둔다.
2. head의exp_data== exp_name
3. 와 같다면
    1. head의 type이 1이라면
       head의 val이 반환값이다.
    2. head의 type이 2이라면
       line은head의 line이다.
       -1 반환값
4. head의exp_data==exp_name
5. 와 같은게 없다면 head는 head의 next를 해준다. head의 next가 NULL이 아닐때까지 반복해준준다.

retVal = 1

getVal()


newnode.type = 1
newnode.exp_data = 'c'
newnode.val = 3
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'b'
newnode.val = 2
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'a'
newnode.val = 1
newnode.line = 0
next
Newnode
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
Newnode(
newnode.type = 2
newnode.exp_data = ' m'
newnode.val = 0
newnode.line = 13
next
Newnode
newnode.type = 2
newnode.exp_data = ' f'
newnode.val = 0
newnode.line = 6
next
Newnode
newnode.type = 2
newnode.exp_data = ' g'
newnode.val = 0
newnode.line = 1
next
Newnode
NULL
newnode.type = 3
newnode.exp_data = ' '
newnode.val = 0
newnode.line = 10
next
newnode.type = 1
newnode.exp_data = c
newnode.val = 2
newnode.line = 0
next
newnode.type = 1
newnode.exp_data = b
newnode.val = 1
newnode.line = 0
next
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
newnode.type = 1
newnode.exp_data = a
newnode.val = 3
newnode.line = 0
next
newnode.type = 2
newnode.exp_data = f
newnode.val = 0
newnode.line = 6
next
newnode.type = 3
newnode.exp_data = ' '
newnode.val = 0
newnode.line = 18
next
reset(f)
top
stack
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
Newnode(3)
newnode.type = 1
newnode.exp_data = x
newnode.val = 3
newnode.line = 0
next
newnode.type = 2
newnode.exp_data = g
newnode.val = 0
newnode.line = 1
next
reset g
head

(1+2-3+x);

```
//만일 retVal이 -1이 아니면서 -999이 아니라면
if ((retVal!=-1) & (retVal!=-999))
{
    //postfix[y]에 retVal+48을 한다.
    postfix[y]=retVal+48;
    //y에 1증가
    y++;

}
```

postfix[5] = 51

(1+2-3+x);

세분화2-4-1-1 (begin과 end가 아니면) - '('라면 세분화 ⓐ

')'이라면

```
else if (lineyedek[i]==')')
                {
                    //(3)-3-2-1
                    ////0이면 true 1이면 false

                    if (!isStackEmpty(MathStack) != 0 ) //0일때 - OpStack->top==0이 아닐 때
                    {
                        // y = 0
                        //Null과 0은 같다
                        postfix[y]=PopOp(MathStack); //
                        y++;
                    }
                }
```

Postfix[6] ='+' Postfix[]

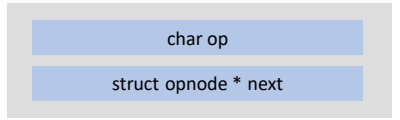
1	2	'+'	3	-	51	+

isStackEmpty

```
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)

1. 만약 stck의 top가 0이라면
   1. 반환값 1
2. 아니면 반환값 0



opNode

PopOp(opstck)

1. opstck의 top이 NULL이면 "Error, empty stack…" 라고 하고 return null;

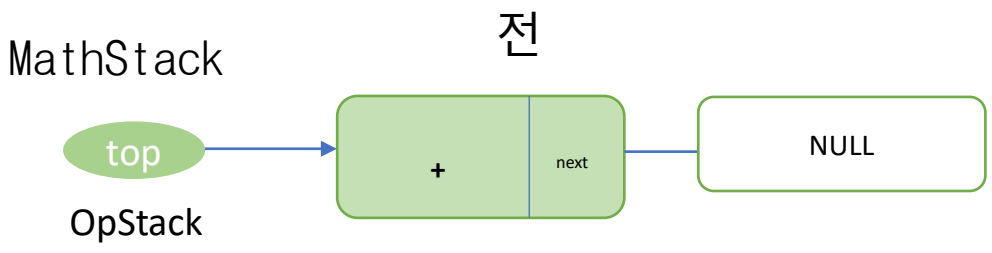
opstck의 top이 NULL이 아니면,

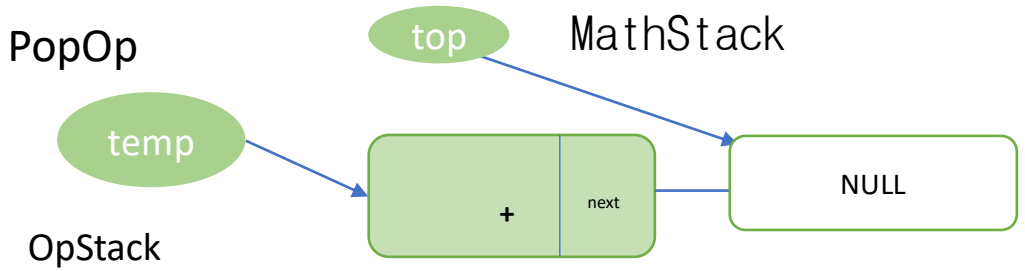
1. op =+(opstck의 top의 op)를 가리킨다.
2. temp는 opstck의 top이다.
3. opstck의 top은opstck의 top의 next이다.
4. temp를 메모리 해체를한다.
5. return은 +

```
char PopOp(OpStack *opstck)
{
opNode *temp;
char op;
if (opstck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
op=opstck->top->op;
temp=opstck->top;
opstck->top=opstck->top->next;
free(temp);
return op;
}
return NULL;
}
```

Op = +

메모리 해제





```c
if (WillBreak==0)
                {
                /* get out items left in the mathstack */
                while (isStackEmpty(MathStack)==0)
                {

                    /* add the poped operator to the postfix */
                    postfix[y]=PopOp(MathStack);
                    y++;
                }

                postfix[y]='\0';

                //MathStack=FreeAll(MathStack);

                /* now calculate the postfix */
                /*printf("\nCURRENT POSTFIX=%s\n",postfix);*/

                i=0;

                CalcStack->top=NULL;
                while(postfix[i]!='\x0')
                {
                    if (isdigit(postfix[i])) {
                        /* push to stack */
                        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
                    }
                    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
                    {
                        val1=PopPostfix(CalcStack);

                        val2=PopPostfix(CalcStack);


                        switch (postfix[i])
                        {
                            case '+': resultVal=val2+val1;break;
                            case '-': resultVal=val2-val1;break;
                            case '/': resultVal=val2/val1;break;
                            case '*': resultVal=val2*val1;break;
                        }

                        CalcStack=PushPostfix(resultVal,CalcStack);
                    }
                    i++;
                }

                //CalcStack=FreeAll(CalcStack);
                LastExpReturn=CalcStack->top->val;


                }
```

①

isStackEmpty

```c
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)

1. 만약 stck의 top가 0이라면
    1. 반환값 1
2. 아니면 반환값 0

ⓐ

```
postfix[y]='\0';

//MathStack=FreeAll(MathStack);

/* now calculate the postfix */
/*printf("\nCURRENT POSTFIX=%s\n",postfix);*/

i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
        val1=PopPostfix(CalcStack);

        val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

Postfix[i]이 \0일때까지 반복실행

Postfix[i]가 숫자일시,

'0'은 아스키코드 48

1증가

Postfix[]

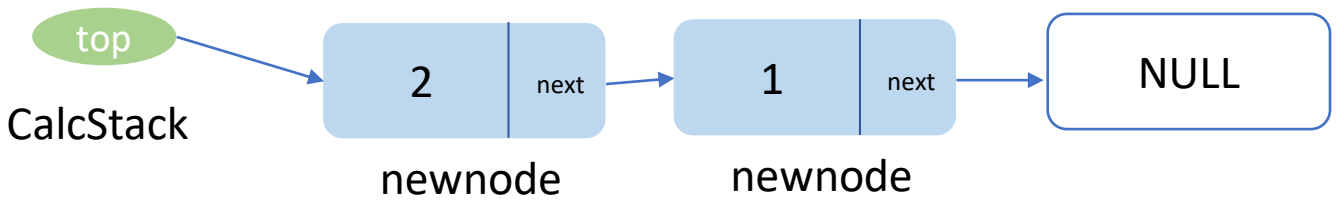
ⓐ

1	2	'+'	3	-	51	+	\0

```
PostfixStack * PushPostfix(int val,PostfixStack *poststck)
{
Postfixnode *newnode;
if ((newnode=(Postfixnode*)malloc(sizeof(Postfixnode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->val=val;
newnode->next=poststck->top;
poststck->top=newnode;
return poststck;
}
}
```

PostfixStack * PushPostfix(int val,PostfixStack *poststck)

1. Postfixnode구조체의 자료형인 newnode 포인터 선언
2. 만약 newnode는 Postfixnode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   NULL반환
3. 아니면라면,
   1. newnode의 val 은 매개변수 val이다
   2. newnode의 next 는 poststck의 top이다.
   3. poststck의 top 는 newnode이다.
   4. 반환값 poststck



```
i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
①       val1=PopPostfix(CalcStack);

②       val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

val1 = 2
val2=1

resultVal = 1+2

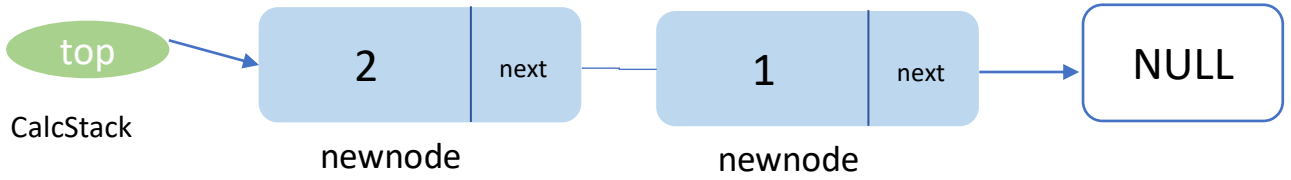
Postfix[]

1	2	'+'	3	-	51	+	\0

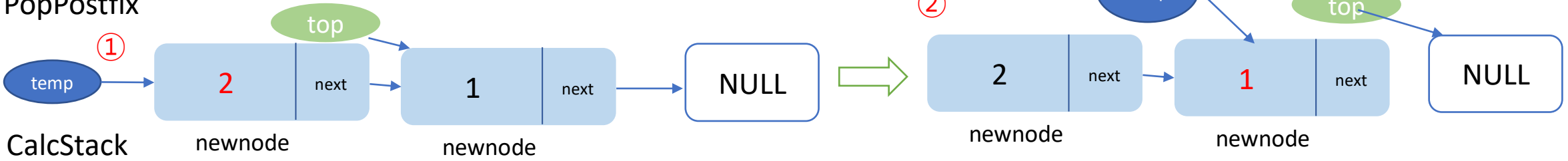
```
char PopPostfix(PostfixStack *poststck)
{
Postfixnode *temp;
int val;
if (poststck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
val=poststck->top->val;
temp=poststck->top;
poststck->top=poststck->top->next;
free(temp);
return val;
}
return NULL;
}
```

char PopPostfix(PstfixStack *poststcPostfixnode구조체의 자료k)

1. 형 temp 포인터 선언
2. 정수자료형인 val선언
3. 만약 poststck의 top이 NULL이라면
   1. ERROR, empty stack...console에 출력
4. 아니라면,
   1. val 은 poststck의 top의 val이다.
   2. tmep는 poststck의 top
   3. poststck의 top 은 poststck의 top의 next이다.
   4. temp의 동적메모리 해제
   5. 반환값 val
5. 반환값 NULL



PopPostfix



```
i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
        val1=PopPostfix(CalcStack);

        val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

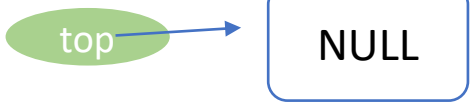
pushPostfix(3,CalcStck)

```
PostfixStack * PushPostfix(int val,PostfixStack *poststck)
{
Postfixnode *newnode;
if ((newnode=(Postfixnode*)malloc(sizeof(Postfixnode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->val=val;
newnode->next=poststck->top;
poststck->top=newnode;
return poststck;
}
}
```

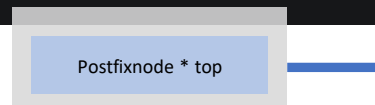
PostfixStack * PushPostfix(int val,PostfixStack *poststck)

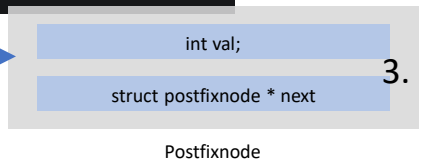
1. Postfixnode구조체의 자료형인 newnode 포인터 선언
2. 만약 newnode는 Postfixnode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   NULL반환
3. 아니면라면,
   1. newnode의 val 은 매개변수 val이다
   2. newnode의 next 는 poststck의 top이다.
   3. poststck의 top 는 newnode이다.
   4. 반환값 poststck

pushPostfix 전



CalcStack





구조체 정리

pushPostfix 후

CalcStack

newnode

Postfix[]

1	2	'+'	3	-	51	+	\0

```
postfix[y]='\0';

//MathStack=FreeAll(MathStack);

/* now calculate the postfix */
/*printf("\nCURRENT POSTFIX=%s\n",postfix);*/

i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
        val1=PopPostfix(CalcStack);

        val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

Postfix[i]이 \0일때까지 반복실행

Postfix[i]가 숫자일시,

'0'은 아스키코드 48

1증가

Postfix[]

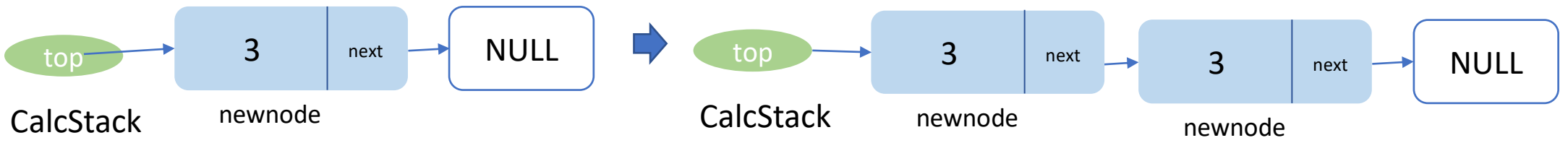
1	2	'+'	3	-	51	+	\0

```
{
Postfixnode *newnode;
if ((newnode=(Postfixnode*)malloc(sizeof(Postfixnode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->val=val;
newnode->next=poststck->top;
poststck->top=newnode;
return poststck;
}
}
```

PostfixStack * PushPostfix(int val,PostfixStack *poststck)

1. Postfixnode구조체의 자료형인 newnode 포인터 선언
2. 만약 newnode는 Postfixnode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   NULL반환
3. 아니면라면,
   1. newnode의 val 은 3 이다
   2. newnode의 next 는 poststck의 top이다.
   3. poststck의 top 는 newnode이다.
   4. 반환값 poststck

PushPostfix()



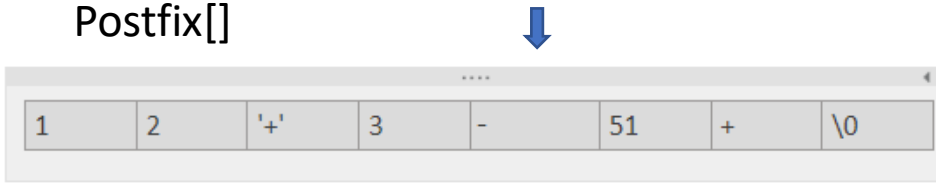
```
i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
    ① val1=PopPostfix(CalcStack);

    ② val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

val1 = 3
val2=3

resultVal =3-3

pushPostfix(0,CalcStck)

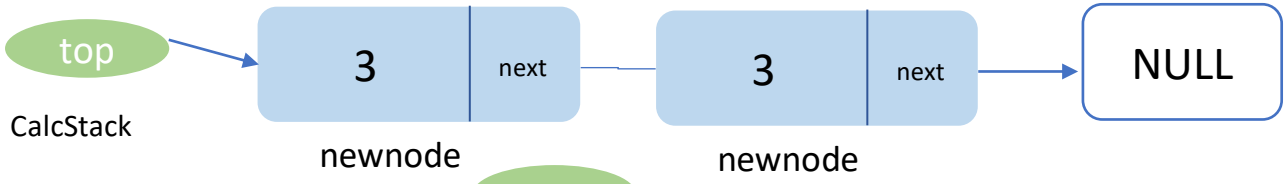
Postfix[]



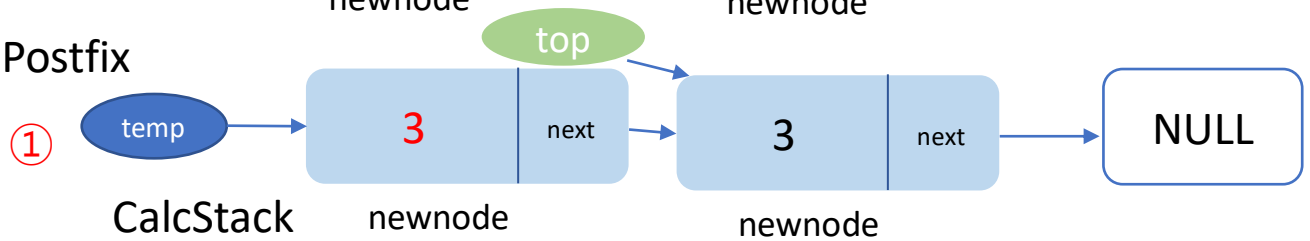

```
char PopPostfix(PostfixStack *poststck)
{
Postfixnode *temp;
int val;
if (poststck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
val=poststck->top->val;
temp=poststck->top;
poststck->top=poststck->top->next;
free(temp);
return val;
}
return NULL;
}
```

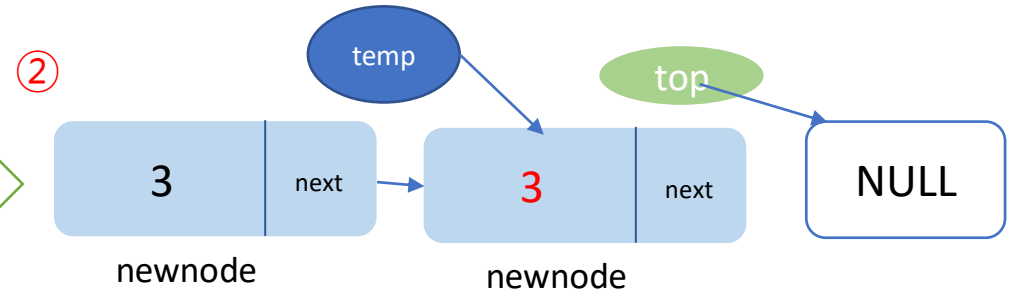
char PopPostfix(PstfixStack *poststcPostfixnode구조체의 자료k)

1. 형 temp 포인터 선언
2. 정수자료형인 val선언
3. 만약 poststck의 top이 NULL이라면
   1. ERROR, empty stack...console에 출력
4. 아니라면,
   1. val 은 poststck의 top의 val이다.
   2. tmep는 poststck의 top
   3. poststck의 top 은 poststck의 top의 next이다.
   4. temp의 동적메모리 해제
   5. 반환값 val
5. 반환값 NULL



PopPostfix





Postfix[]

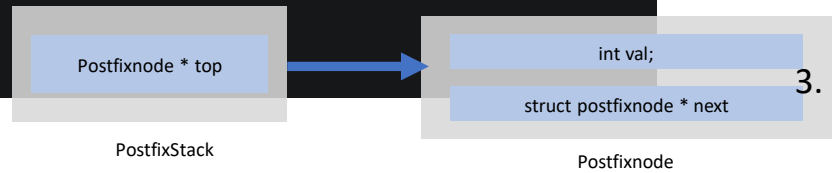
```
i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
        val1=PopPostfix(CalcStack);

        val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

```
PostfixStack * PushPostfix(int val,PostfixStack *poststck)
{
Postfixnode *newnode;
if ((newnode=(Postfixnode*)malloc(sizeof(Postfixnode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->val=val;
newnode->next=poststck->top;
poststck->top=newnode;
return poststck;
}
}
```

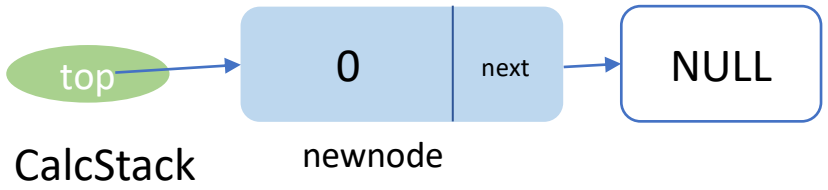
PostfixStack * PushPostfix(int val,PostfixStack *poststck)

1. Postfixnode구조체의 자료형인 newnode 포인터 선언
2. 만약 newnode는 Postfixnode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   NULL반환

3. 아니면라면,
   1. newnode의 val 은 매개변수 val이다
   2. newnode의 next 는 poststck의 top이다.
   3. poststck의 top 는 newnode이다.
   4. 반환값 poststck



구조체 정리



```
postfix[y]='\0';

//MathStack=FreeAll(MathStack);

/* now calculate the postfix */
/*printf("\nCURRENT POSTFIX=%s\n",postfix);*/

i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
        val1=PopPostfix(CalcStack);

        val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

Postfix[i]이 \0일때까지 반복실행

Postfix[i]가 숫자일시,

'0'은 아스키코드 48

1증가

Postfix[]

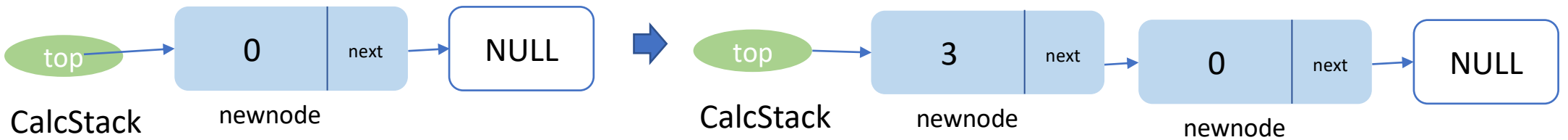
1	2	'+'	3	-	51	+	\0

```
PostfixStack * PushPostfix(int val,PostfixStack *poststck)
{
Postfixnode *newnode;
if ((newnode=(Postfixnode*)malloc(sizeof(Postfixnode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->val=val;
newnode->next=poststck->top;
poststck->top=newnode;
return poststck;
}
}
```

PostfixStack * PushPostfix(int val,PostfixStack *poststck)

1. Postfixnode구조체의 자료형인 newnode 포인터 선언
2. 만약 newnode는 Postfixnode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   NULL반환
3. 아니면라면,
   1. newnode의 val 은 3I 이다
   2. newnode의 next 는 poststck의 top이다.
   3. poststck의 top 는 newnode이다.
   4. 반환값 poststck

PushPostfix()



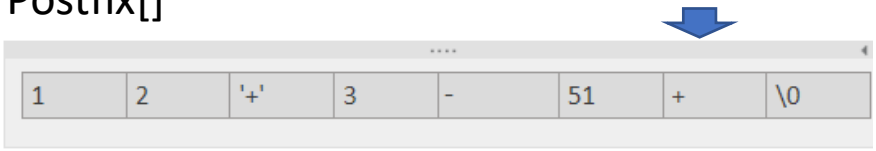
```
i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
    ① val1=PopPostfix(CalcStack);

    ② val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

val1 = 3
val2=0

resultVal =3

pushPostfix(3,CalcStck)

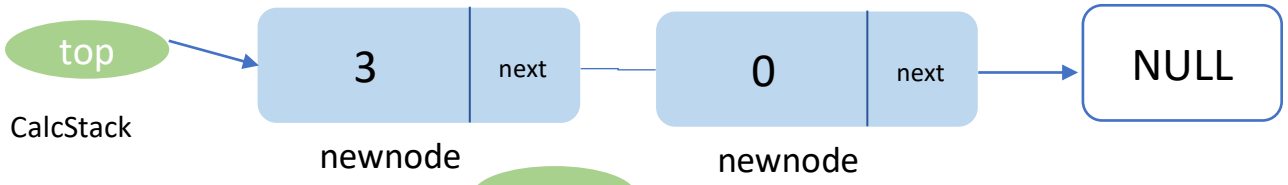
Postfix[]



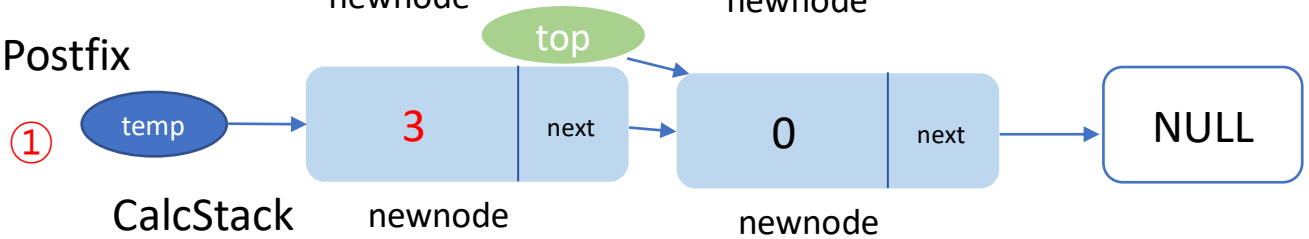

```
char PopPostfix(PostfixStack *poststck)
{
Postfixnode *temp;
int val;
if (poststck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
val=poststck->top->val;
temp=poststck->top;
poststck->top=poststck->top->next;
free(temp);
return val;
}
return NULL;
}
```

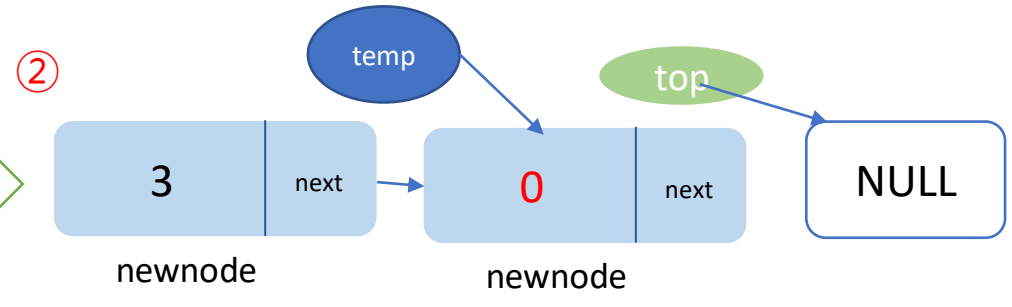
char PopPostfix(PstfixStack *poststcPostfixnode구조체의 자료k)

1. 형 temp 포인터 선언
2. 정수자료형인 val선언
3. 만약 poststck의 top이 NULL이라면
   1. ERROR, empty stack...console에 출력
4. 아니라면,
   1. val 은 poststck의 top의 val이다.
   2. tmep는 poststck의 top
   3. poststck의 top 은 poststck의 top의 next이다.
   4. temp의 동적메모리 해제
   5. 반환값 val
5. 반환값 NULL



PopPostfix





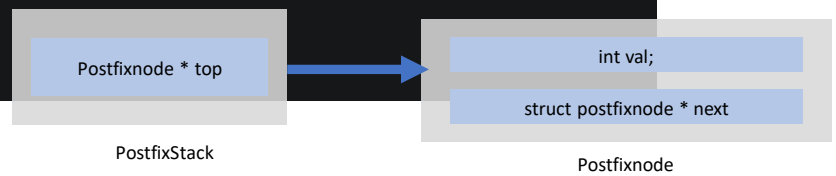
Postfix[]

```
i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
        val1=PopPostfix(CalcStack);

        val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

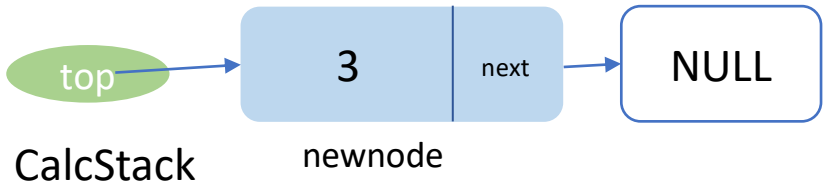
1	2	'+'	3	-	51	+	\0

pushPostfix(1,CalcStck)

```
PostfixStack * PushPostfix(int val,PostfixStack *poststck)
{
Postfixnode *newnode;
if ((newnode=(Postfixnode*)malloc(sizeof(Postfixnode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->val=val;
newnode->next=poststck->top;
poststck->top=newnode;
return poststck;
}
}
```



구조체 정리



LastExpReturn = 3

PostfixStack * PushPostfix(int val,PostfixStack *poststck)

1. Postfixnode구조체의 자료형인 newnode 포인터 선언
2. 만약 newnode는 Postfixnode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   NULL반환
3. 아니면라면,
   1. newnode의 val 은 매개변수 val이다
   2. newnode의 next 는 poststck의 top이다.
   3. poststck의 top 는 newnode이다.
   4. 반환값 poststck

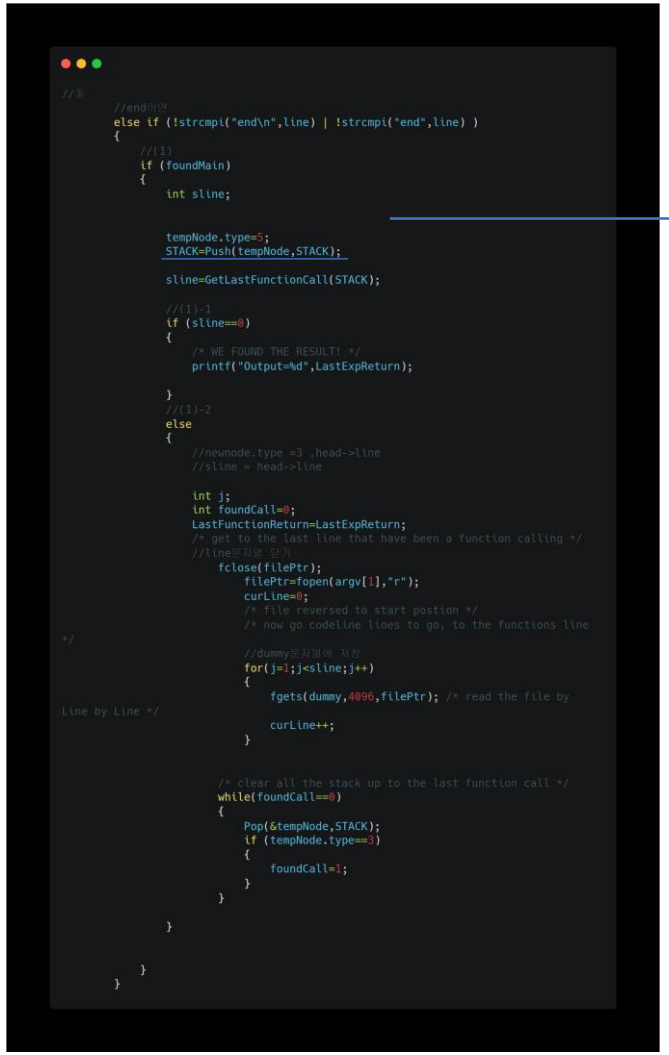
파일반복

```
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //fget을 통해 function f(int a)을 읽어 line에 저장 .
    fgets(line,4096,filePtr); /* read the file by Line by Line */
    /* scan for /t characters. get rid of them! */

    // whil문 line[k]이  null 이 아닐 시때까지 반복.
    while(line[k]!='₩0')
    {
       //  4-1) 만약 line[k]가 ₩t(탭키)일 시,
     if (line[k]=='₩t')
     {
       //line[k]는  ' '으로 바꾼다.
      line[k]=' ';
     }
      //k에 1증가
     k++;
    }
    //5. line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을  ' '로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  둘이 동일하면 1(true) 반환하고 동일하지 않으면 0(false)
    //begin과 line이 동일하다면
    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }


    //ⓑ
    //  //strcmpi :  둘이 동일하면 1(true) 반환하고 동일하지 않으면 0(false)
    //line이 end라면
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }

    //ⓒ begin과 end가 아니면
    else{ ... }
```

input 2.sql (읽을 파일)

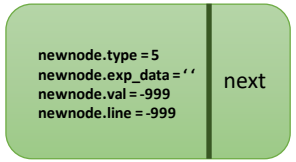
```
function g(int x)
begin
 (1+2-3+x);
end

function f(int a)
begin
  int b = 1;
  int c = 2;
  ((b*c)+g(a));
end

function main()
begin
  int a = 1;
  int b = 2;
  int c = 3;
  ((2 + f(c) ) * a);
end
```

line이 end일 때



```
//8
        //end이면
        else if (!strcmpi("end\n",line) | !strcmpi("end",line) )
        {
            //(1)
            if (foundMain)
            {
                int sline;


                tempNode.type=5;
                STACK=Push(tempNode,STACK);

                sline=GetLastFunctionCall(STACK);

                //(1)-1
                if (sline==0)
                {
                    /* WE FOUND THE RESULT! */
                    printf("Output=%d",LastExpReturn);

                }
                //(1)-2
                else
                {
                    //newnode.type =3 ,head->line
                    //sline = head->line

                    int j;
                    int foundCall=0;
                    LastFunctionReturn=LastExpReturn;
                    /* get to the last line that have been a function calling */
                    //line문자열 담기
                        fclose(filePtr);
                            filePtr=fopen(argv[1],"r");
                            curLine=0;
                            /* file reversed to start postion */
                            /* now go codeline lines to go, to the functions line
*/
                            //dummy문자열에 저장
                            for(j=1;j<sline;j++)
                            {
                                fgets(dummy,4096,filePtr); /* read the file by
Line by Line */
                                curLine++;
                            }


                            /* clear all the stack up to the last function call */
                        while(foundCall==0)
                        {
                            Pop(&tempNode,STACK);
                            if (tempNode.type==3)
                            {
                                foundCall=1;
                            }
                        }

                }


            }
        }
```

```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```



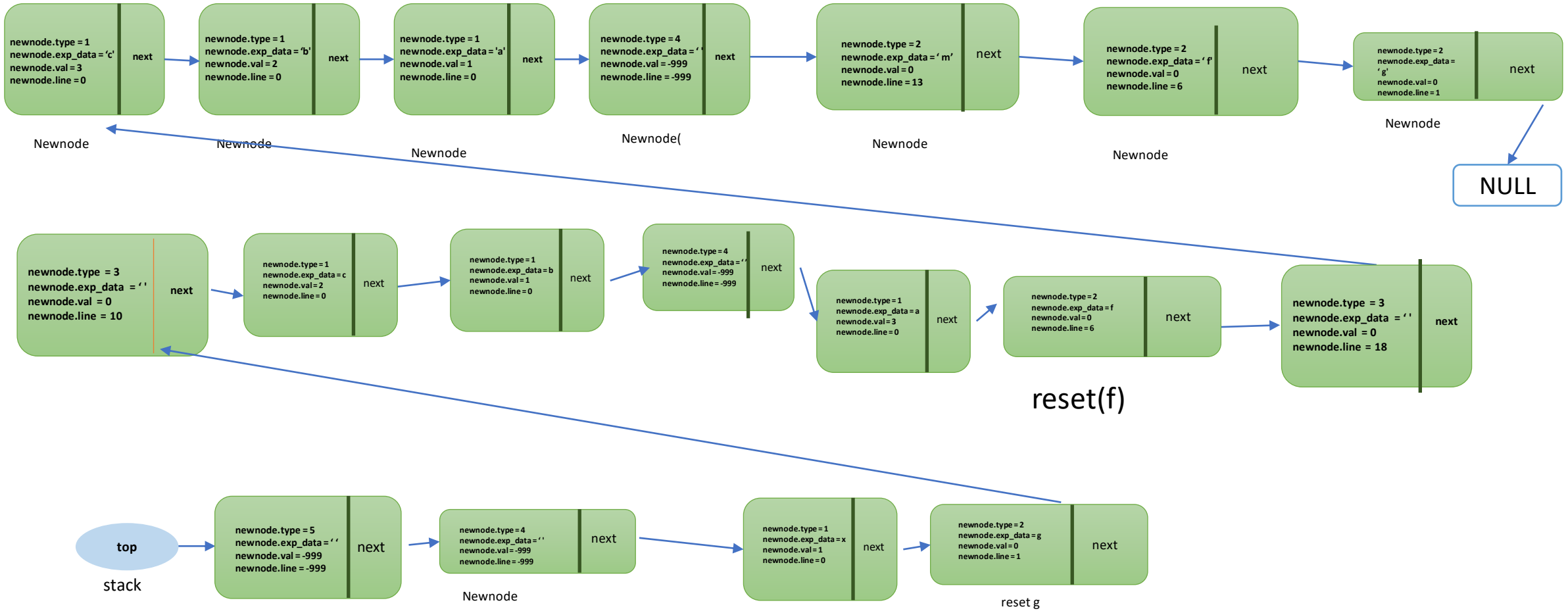
Stack * Push(Node sNode,Stack *stck)

1. Node 자료형의 newnode포인터선언;
2. 만약 newnode는 Node타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   return NULL

3.아니라면

newnode의 type은 5의 타입이 된다.
newnode 의 val은 -999 이된다.
newnode 의 line 은 -999 이 된다.
newnode 의 next 는 stck의 top이 된다.
stck의 top은 newnode이다

반환값 stck


newnode.type = 1
newnode.exp_data = 'c'
newnode.val = 3
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'b'
newnode.val = 2
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'a'
newnode.val = 1
newnode.line = 0
next
Newnode
newnode.type = 4
newnode.exp_data = ''
newnode.val = -999
newnode.line = -999
next
Newnode(
newnode.type = 2
newnode.exp_data = 'm'
newnode.val = 0
newnode.line = 13
next
Newnode
newnode.type = 2
newnode.exp_data = 'f'
newnode.val = 0
newnode.line = 6
next
Newnode
newnode.type = 2
newnode.exp_data = 'g'
newnode.val = 0
newnode.line = 1
next
Newnode
NULL
newnode.type = 3
newnode.exp_data = ''
newnode.val = 0
newnode.line = 10
next
newnode.type = 1
newnode.exp_data = c
newnode.val = 2
newnode.line = 0
next
newnode.type = 1
newnode.exp_data = b
newnode.val = 1
newnode.line = 0
next
newnode.type = 4
newnode.exp_data = ''
newnode.val = -999
newnode.line = -999
next
newnode.type = 1
newnode.exp_data = a
newnode.val = 3
newnode.line = 0
next
newnode.type = 2
newnode.exp_data = f
newnode.val = 0
newnode.line = 6
next
reset(f)
newnode.type = 3
newnode.exp_data = ''
newnode.val = 0
newnode.line = 18
next
top
stack
newnode.type = 5
newnode.exp_data = ''
newnode.val = -999
newnode.line = -999
next
newnode.type = 4
newnode.exp_data = ''
newnode.val = -999
newnode.line = -999
next
Newnode
newnode.type = 1
newnode.exp_data = x
newnode.val = 1
newnode.line = 0
next
newnode.type = 2
newnode.exp_data = g
newnode.val = 0
newnode.line = 1
next
reset g

line이 end일 때

sline=10

GetLastFunctionCall

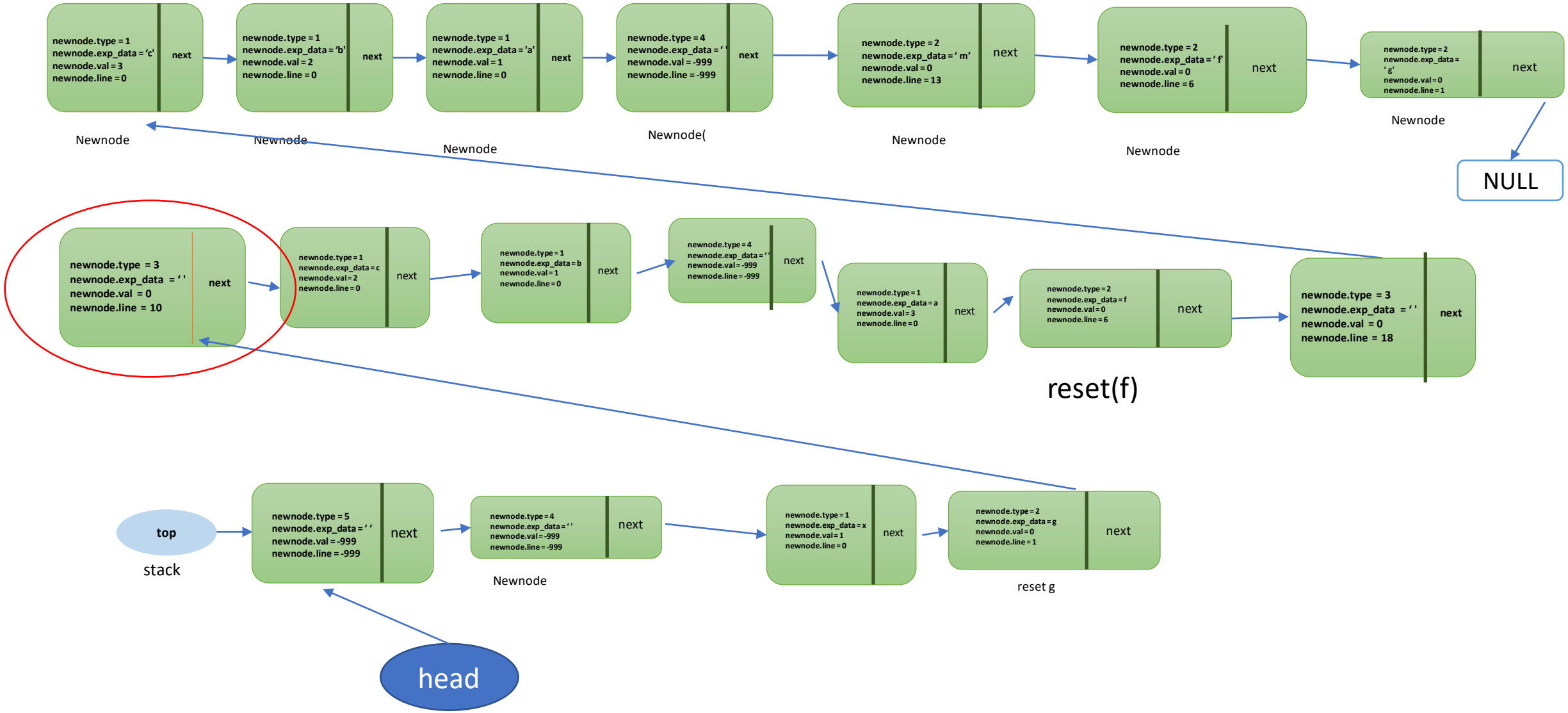
```
int GetLastFunctionCall(Stack *stck)
{
Node * head;

  if (stck->top == NULL )
  {
    printf("ERROR, empty stack...");
  }
  else
  {
    head=stck->top;
    do
    {
      if ( head->type==3 )
      {
        return head->line;
      }
      else
      {
        head=head->next;
      }
    } while (head->next!=NULL);
  }

  return 0;



}
```

10

int GetLastFunctionCall(Stack *stck)

1. Node의 구조체의 자료형인 포인터 head
2. 만약 stck의 top 이 NULL이라면,
    1. ERROR, empty stack… 이라고 콘솔 출력
3. 아니라면,
    1. head는 stck의 top이다.
    2. do
        1. head의 type은 3이라면
            1. 반환값 head의 line
        2. 아니라면
            1. head는 head의 next이다.
    3. while (head의 next가 NULL이 아니라면 반복)
4. 반환값 0

GetLastFunctionCall



```
//(1)-2
else
{
    //newnode.type =3 ,head->line
    //sline = head->line

    int j;
    int foundCall=0;
    LastFunctionReturn=LastExpReturn;
    /* get to the last line that have been a function calling */
    //filePtr 닫기
        fclose(filePtr);
        /// filePtr= argv[1] 파일 읽기모드는 파일열기
            filePtr=fopen(argv[1],"r");
            curLine=0;
            /* file reversed to start postion */
            /* now go codeline lines to go, to the functions line */
            //dummy문자열에 저장
            for(j=1;j<sline;j++)
            {
                //
                fgets(dummy,4096,filePtr); /* read the file by Line by Line */
                curLine++;
            }


        /* clear all the stack up to the last function call */
        while(foundCall==0)
        {
            Pop(&tempNode,STACK);
            if (tempNode.type==3)
            {
                foundCall=1;
            }
        }

}
```

LastExpReturn = 3

LastFunctionReturn= 3

```
curLine=0;
sline=10
for문
//dummy문자열에 저장
for(j=1;j<10;j++)
{
//
fgets(dummy,4096,filePtr);
curLine++;//9
}
```

input 2.sql (읽을 파일)

```
function g(int x)
begin
 (1+2-3+x);
end

function f(int a)
begin
  int b = 1;
  int c = 2;
  ((b*c)+g(a));
end

function main()
begin
  int a = 1;
  int b = 2;
  int c = 3;
  ((2 + f(c) ) * a);
end
```

```
        }

    /* clear all the stack up to the last function call */
    while(foundCall==0)
    {
        Pop(&tempNode,STACK);
        if (tempNode.type==3)
        {
            foundCall=1;
        }
    }

}
```

tempNode.type=3이나오면
foundCall=1로 변경되면while문 종료

Pop

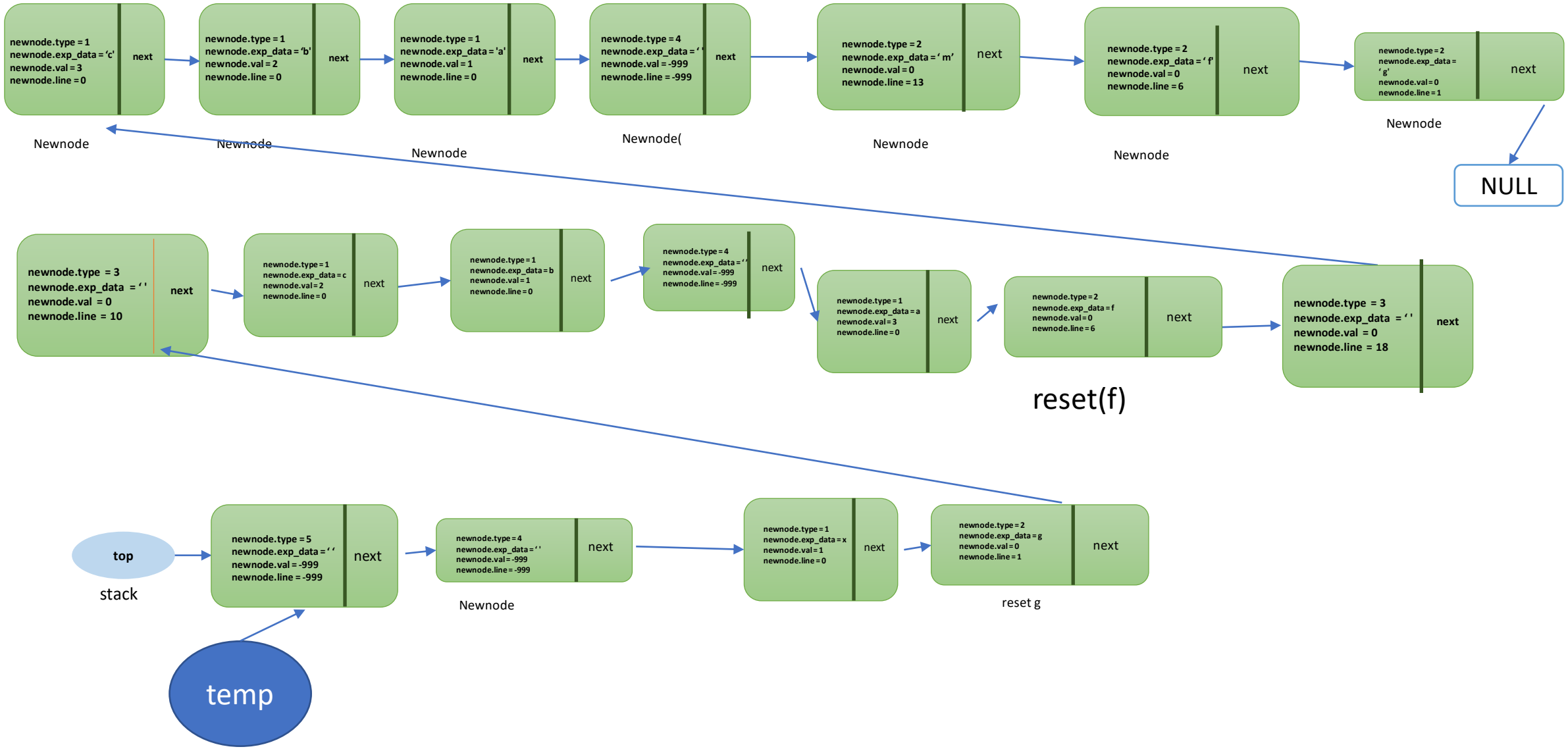
```
void Pop(Node * sNode,Stack *stck)
{
Node *temp;

if (stck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
sNode->exp_data=stck->top->exp_data;
sNode->type=stck->top->type;
sNode->line=stck->top->line;
sNode->val=stck->top->val;
temp=stck->top;
stck->top=stck->top->next;
free(temp);
}
}
```

void Pop(Node * sNode,Stack *stck)

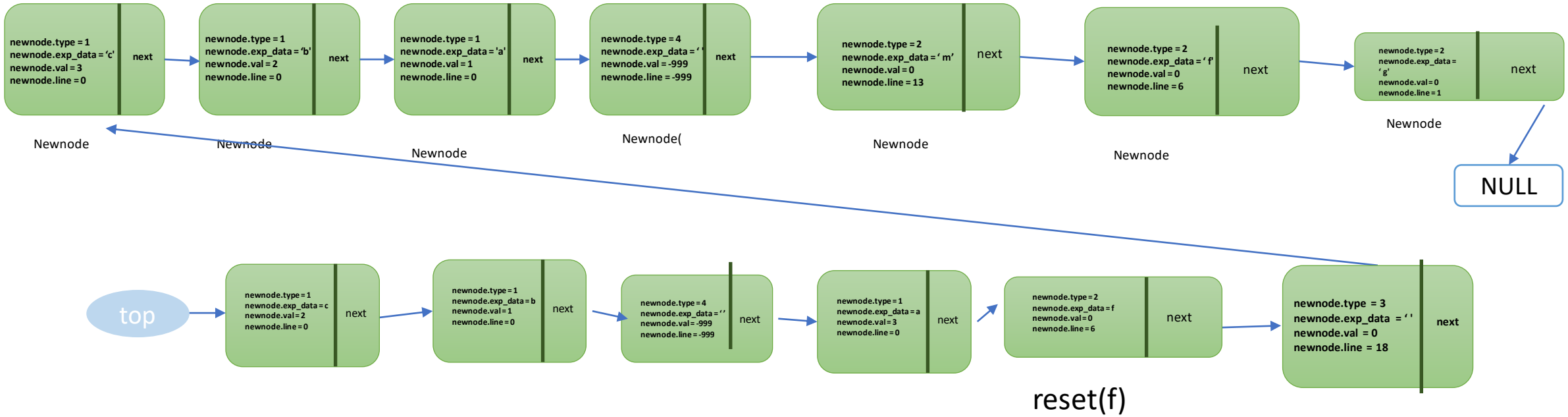
1. Node구조체 자료형인 temp 포인터 선언
2. 만약 stck의 top 이 NULL이라면,
   1. ERROR, empty stack... console출력
3. 아니라면
   1. sNode의 exp_data 는 stck의 top의 exp_data이다.
   2. sNode의 type는 stck의 top의 type 이다
   3. sNode의 line는 stck의 top의 line이다
   4. sNode의 val는 stck의 top의 val이다
   5. temp는 stck의 top이다.
   6. stck의 top은 stck의 top의 next이다.
   7. temp동적메모리 해제

newnode.type = 5
newnode.exp_data = ‘‘
newnode.val = -999
newnode.line = -999

Pop() 전


newnode.type = 1
newnode.exp_data = 'c'
newnode.val = 3
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'b'
newnode.val = 2
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'a'
newnode.val = 1
newnode.line = 0
next
Newnode
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
Newnode(
newnode.type = 2
newnode.exp_data = ' m'
newnode.val = 0
newnode.line = 13
next
Newnode
newnode.type = 2
newnode.exp_data = ' f'
newnode.val = 0
newnode.line = 6
next
Newnode
newnode.type = 2
newnode.exp_data = ' g'
newnode.val = 0
newnode.line = 1
next
Newnode
NULL
newnode.type = 3
newnode.exp_data = ' '
newnode.val = 0
newnode.line = 10
next
newnode.type = 1
newnode.exp_data = c
newnode.val = 2
newnode.line = 0
next
newnode.type = 1
newnode.exp_data = b
newnode.val = 1
newnode.line = 0
next
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
newnode.type = 1
newnode.exp_data = a
newnode.val = 3
newnode.line = 0
next
newnode.type = 2
newnode.exp_data = f
newnode.val = 0
newnode.line = 6
next
newnode.type = 3
newnode.exp_data = ' '
newnode.val = 0
newnode.line = 18
next
reset(f)
top
stack
newnode.type = 5
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
Newnode
newnode.type = 1
newnode.exp_data = x
newnode.val = 1
newnode.line = 0
next
newnode.type = 2
newnode.exp_data = g
newnode.val = 0
newnode.line = 1
next
reset g
temp

Pop() 후


newnode.type = 1
newnode.exp_data = 'c'
newnode.val = 3
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'b'
newnode.val = 2
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'a'
newnode.val = 1
newnode.line = 0
next
Newnode
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
Newnode(
newnode.type = 2
newnode.exp_data = ' m'
newnode.val = 0
newnode.line = 13
next
Newnode
newnode.type = 2
newnode.exp_data = ' f'
newnode.val = 0
newnode.line = 6
next
Newnode
newnode.type = 2
newnode.exp_data = ' g'
newnode.val = 0
newnode.line = 1
next
Newnode
NULL
top
newnode.type = 1
newnode.exp_data = c
newnode.val = 2
newnode.line = 0
next
newnode.type = 1
newnode.exp_data = b
newnode.val = 1
newnode.line = 0
next
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
newnode.type = 1
newnode.exp_data = a
newnode.val = 3
newnode.line = 0
next
newnode.type = 2
newnode.exp_data = f
newnode.val = 0
newnode.line = 6
next
newnode.type = 3
newnode.exp_data = ' '
newnode.val = 0
newnode.line = 18
next
reset(f)

((b*c)+g(a));

```
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //fget을 통해 function f(int a)을 읽어 line에 저장 .
    fgets(line,4096,filePtr); /* read the file by Line by Line */
    /* scan for /t characters. get rid of them! */

    // whil문 line[k]이  null 이 아닐 시때까지 반복.
    while(line[k]!='₩0')
    {
        //  4-1) 만약 line[k]가 ₩t(탭키)일 시,
    if (line[k]=='₩t')
    {
        //line[k]는  ' '으로 바꾼다.
      line[k]=' ';
    }
     //k에 1증가
    k++;
    }
    //5. line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을  ' '로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  들이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //begin과 line이 동일하다면
    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }


    //ⓑ
    //  //strcmpi :  들이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //line이 end라면
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }

    //ⓒ begin과 end가 아니면
    else{ ... }
```

input 2.sql (읽을 파일)

```
function g(int x)
begin
  (1+2-3+x);
end

function f(int a)
begin
   int b = 1;
   int c = 2;
   ((b*c)+g(a));
end

function main()
begin
   int a = 1;
   int b = 2;
   int c = 3;
   ((2 + f(c) ) * a);
end
```

((b*c)+g(a));

```
//lineyedek[i] 영어라면
else if (isalpha(lineyedek[i])>0)
{
    int codeline=0;
    int dummyint=0;
    /*look if it's a variable or function call
    int retVal=0;
    retVal=GetVal(lineyedek[i],&codeline,STACK);
    //int codeline =1

    //-1


    ////(3)-3-4-1
    if ((retVal!=-1) & (retVal!=-999)){ ... }
    ////(3)-3-4-2
    //-1
    else{ ... }
}
```

GetVal (b,&codeline,STACK)

GetVal

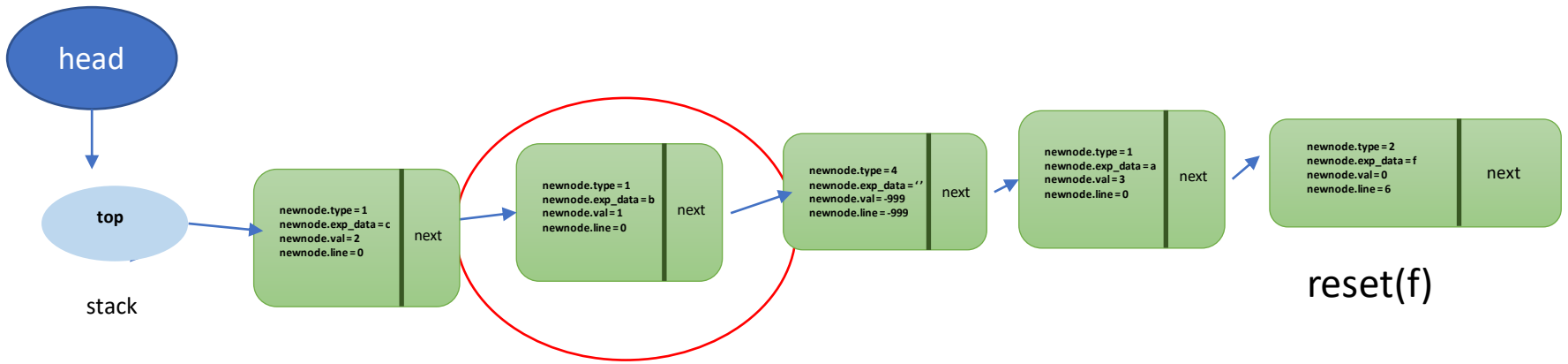
```
int GetVal(char exp_name,int * line,Stack *stck)
{

Node * head;
*line=0;
    if (stck->top == NULL )
    {
        printf("ERROR, empty stack...");
    }
    else
    {
        head=stck->top;
        do
        {
            if ( head->exp_data==exp_name )
            {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }

            }
            else
            {
                head=head->next;
            }
        } while (head->next!=NULL);
        /* check agin once more */
        if (head->exp_data==exp_name)
        {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }


        }
    }

    return -999;
}
```

head->exp_data = b

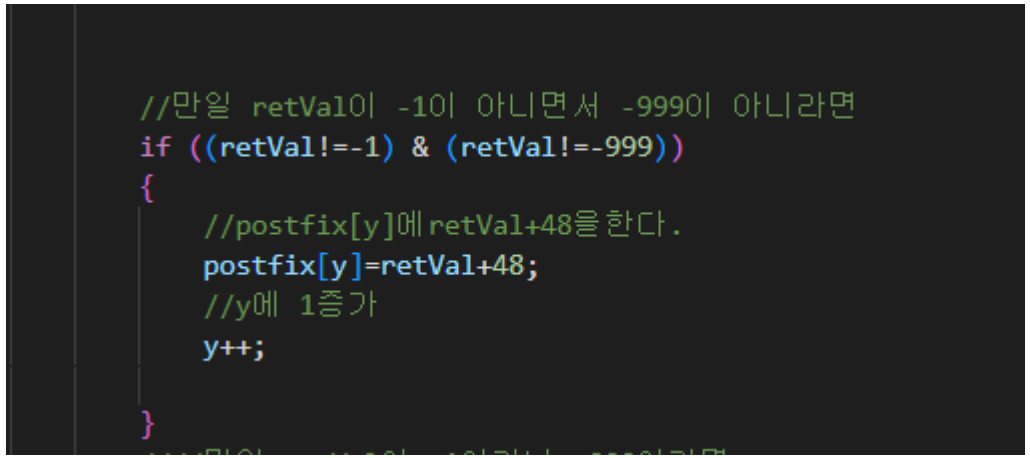
return 1

GetVal()
자료형인 Node인 포인트 head선언

1. head를 stck의 top으로 둔다.
2. head의exp_data==’b’와 같다면
    1. head의 type이 1이라면
       head의 val이 반환값이다.
    2. head의 type이 2이라면
       line은head의 line이다.
       -1 반환값
3. head의exp_data==’b’와 같은게 없다면 head는 head의 next를 해준다. head의 next가 NULL이 아닐때까지 반복해준준다.



((b*c)+g(a));



```
//만일 retVal이 -1이 아니면서 -999이 아니라면
if ((retVal!=-1) & (retVal!=-999))
{
    //postfix[y]에 retVal+48을 한다.
    postfix[y]=retVal+48;
    //y에 1증가
    y++;

}
```

retVal = 1

postfix[0] = 49

((b*c)+g(a));

```
////(3)-3-3
//lineyedek[i]이 '+'이거나 lineyedek[i]이 '-' 이거나 lineyedek[i]이 '+' 이거나 lineyedek[i]이 '/'이라면
else if ((lineyedek[i]=='+') | (lineyedek[i]=='-') | (lineyedek[i]=='*') | (lineyedek[i]=='/'))
{
   ////(3)-3-3-1
   /*operators*/
   //00이면 false 10이면 true
   //처음일 때
   if (isStackEmpty(MathStack) != 0 ) //1일때 - stck->top==0
   {
/* if stack empty push the operator to stack */
      //+
      MathStack=PushOp(lineyedek[i],MathStack);
   }
   ////(3)-3-3-2

   //0일때 - stck -top = 00이 아닐때
   //처음이 아닐때
   else { ... }
}
```

ⓐ

ⓑ

PushOp(*, MathStack)

isStackEmpty

```
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)

1. 만약 stck의 top가 0이라면
   1. 반환값 1
2. 아니면 반환값 0

PushOp

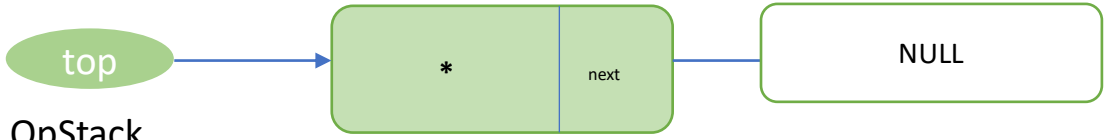
```
OpStack * PushOp(char op,OpStack *opstck)
{
opNode *newnode;
if ((newnode=(opNode*)malloc(sizeof(opNode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->op=op;
newnode->next=opstck->top;
opstck->top=newnode;
return opstck;
}
}
```

OpStack * PushOp(char op,OpStack *opstck)

1. opNode자료형의 newnode포인터선언;
2. 만약 newnode는 opNode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   반환NULL
3. 아니라면,
   1. newnode의 op는 매개변수+이다.
   2. newnode의 next는 opstck의 top이다.
   3. opstck의 top은 newnode이다.

반환값 opstck

PushOp



OpStack
MathStack

((b*c)+g(a));

```
//lineyedek[i] 영어라면
else if (isalpha(lineyedek[i])>0)
{
    int codeline=0;
    int dummyint=0;
    /*look if it's a variable or function call
    int retVal=0;
    retVal=GetVal(lineyedek[i],&codeline,STACK);
    //int codeline =1

    //-1


    ////(3)-3-4-1
    if ((retVal!=-1) & (retVal!=-999)){ ... }
    ////(3)-3-4-2
    //-1
    else{ ... }
}
```

GetVal (c,&codeline,STACK)

GetVal

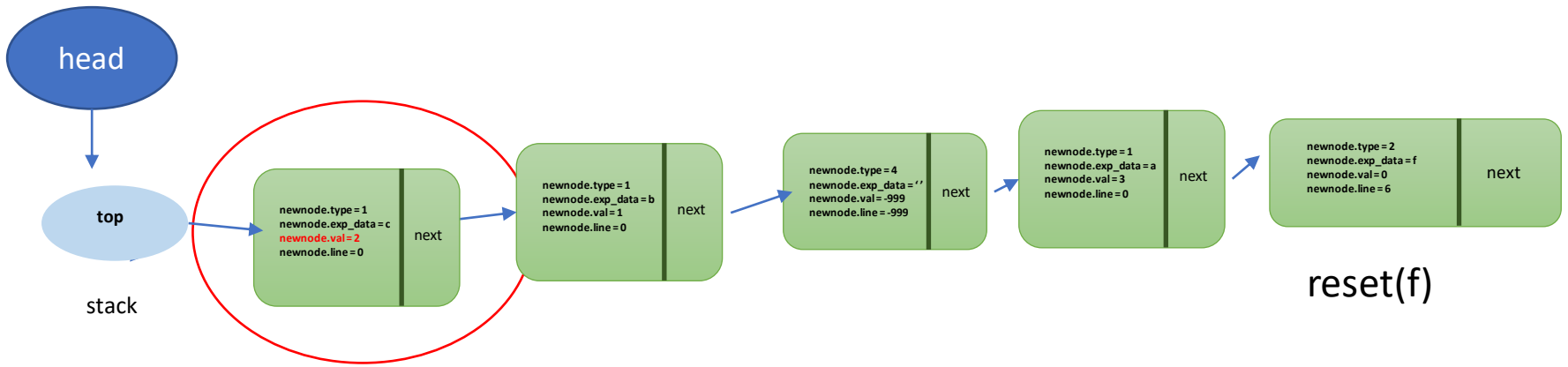
```
int GetVal(char exp_name,int * line,Stack *stck)
{

Node * head;
*line=0;
    if (stck->top == NULL )
    {
        printf("ERROR, empty stack...");
    }
    else
    {
        head=stck->top;
        do
        {
            if ( head->exp_data==exp_name )
            {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }

            }
            else
            {
                head=head->next;
            }
        } while (head->next!=NULL);
        /* check again once more */
        if (head->exp_data==exp_name)
        {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }


        }
    }

    return -999;
}
```

head->exp_data = c

return 2

GetVal()
자료형인 Node인 포인트 head선언

1. head를 stck의 top으로 둔다.
2. head의exp_data==’c’와 같다면
    1. head의 type이 1이라면
       head의 val이 반환값이다.
    2. head의 type이 2이라면
       line은head의 line이다.
       -1 반환값
3. head의exp_data==’c’와 같은게 없다면 head는 head의 next를 해준다. head의 next가 NULL이 아닐때까지 반복해준준다.



((b*c)+g(a));

```
//만일 retVal이 -1이 아니면서 -999이 아니라면
if ((retVal!=-1) & (retVal!=-999))
{
    //postfix[y]에 retVal+48을 한다.
    postfix[y]=retVal+48;
    //y에 1증가
    y++;

}
```

retVal = 2

postfix[1] = 50

((b*c)+g(a));

세분화2-4-1-1 (begin과 end가 아니면) - '('라면 세분화 ⓐ

')'이라면

```
else if (lineyedek[i]==')')
                {
                    //(3)-3-2-1
                    ////0이면 true 1이면 false

                    if (!isStackEmpty(MathStack) != 0 ) //0일때 - OpStack-
>top==0이 아닐 때
                    {
                        // y = 0
                        //Null과 0은 같다
                        postfix[y]=PopOp(MathStack); //
                        y++;
                    }
                }
```

Postfix[2] =‘*’ Postfix[]

49	50	*

isStackEmpty

```
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)

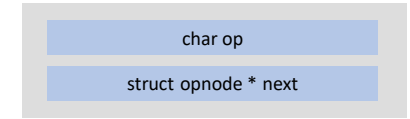
1. 만약 stck의 top가 0이라면
   1. 반환값 1
2. 아니면 반환값 0

PopOp(opstck)

1. opstck의 top이 NULL이면 “Error, empty stack…” 라고 하고 return null;

opstck의 top이 NULL이 아니면,

1. op =*(opstck의 top의 op)를 가리킨다.
2. temp는 opstck의 top이다.
3. opstck의 top은opstck의 top의 next이다.
4. temp를 메모리 해체를한다.
5. return은 *

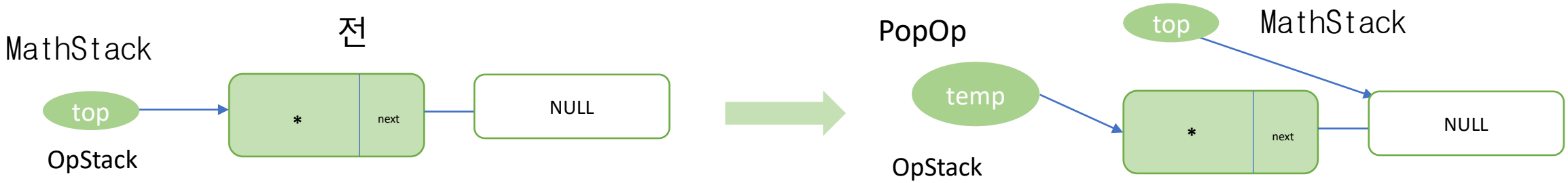


opNode

```
char PopOp(OpStack *opstck)
{
opNode *temp;
char op;
if (opstck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
op=opstck->top->op;
temp=opstck->top;
opstck->top=opstck->top->next;
free(temp);
return op;
}
return NULL;
}
```

Op = +

메모리 해제



((b*c)+g(a));

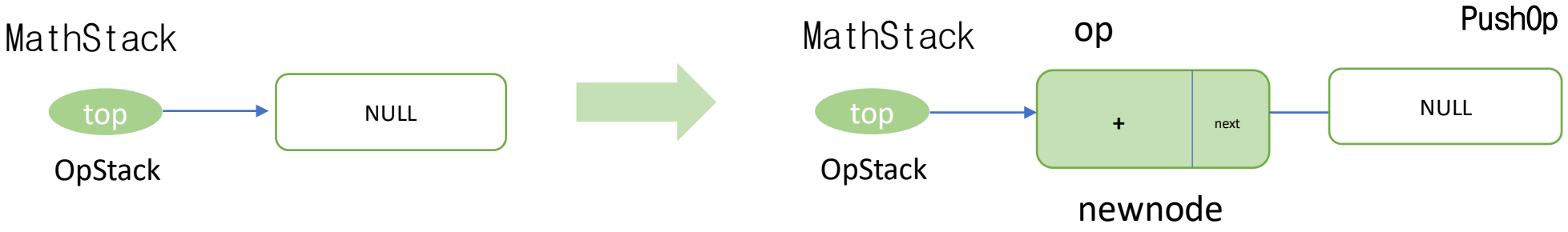
```
    else if ((lineyedek[i]=='+') | (lineyedek[i]=='-') | (lineyedek[i]=='*') |
(lineyedek[i]=='/'))
                {
                    ////(3)-3-3-1
                    /*operators*/
                    //0이면 false 1이면 true
                    //처음일 때
                    if (isStackEmpty(MathStack) != 0 ) //1일때 - stck->top==0
                    {
                /* if stack empty push the operator to stack */

                    MathStack=PushOp(lineyedek[i],MathStack);
                    }
                    ////(3)-3-
```

PsushOp('+' , MathStack)

```
OpStack * PushOp(char op,OpStack *opstck)
{
opNode *newnode;
if ((newnode=(opNode*)malloc(sizeof(opNode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->op=op;
newnode->next=opstck->top;
opstck->top=newnode;
return opstck;
}
}
```

PushOp()
만약 opNode의 구조체 크기가 Null일 경우,
"ERROR, Couldn't allocate memory..."출력 후 NULL 리턴

아니라면,
1. +는 새로운 노드의 op가 된다.
2. opstck->top은 새로운 노드의 next이다
3. newnode는 Opstck의 top은 가리킨다.
return opstck;



((b*c)+g(a));

```
//lineyedek[i] 영어라면
else if (isalpha(lineyedek[i])>0)
{
    int codeline=0;
    int dummyint=0;
    /*look if it's a variable or function call
    int retVal=0;
    retVal=GetVal(lineyedek[i],&codeline,STACK);
    //int codeline =1

    //-1


    ////(3)-3-4-1
    if ((retVal!=-1) & (retVal!=-999)){ ... }
    ////(3)-3-4-2
    //-1
    else{ ... }
}
```

GetVal (g,&codeline,STACK)

GetVal

```
int GetVal(char exp_name,int * line,Stack *stck)
{

Node * head;
*line=0;
    if (stck->top == NULL )
    {
        printf("ERROR, empty stack...");
    }
    else
    {
        head=stck->top;
        do
        {
            if ( head->exp_data==exp_name )
            {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }

            }
            else
            {
                head=head->next;
            }
        } while (head->next!=NULL);
        /* check agin once more */
        if (head->exp_data==exp_name)
        {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }


        }
    }

    return -999;
}
```

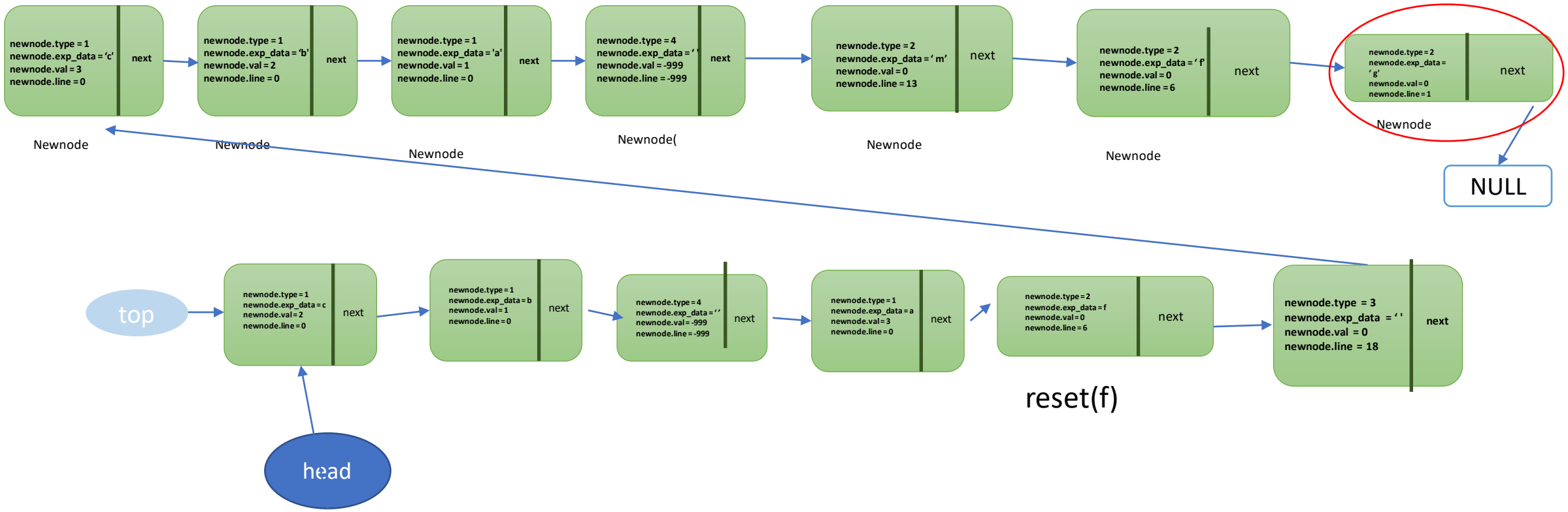
head->exp_data = g

*line = 1
return -1

GetVal()
자료형인 Node인 포인트 head선언

1. head를 stck의 top으로 둔다.
2. head의exp_data== exp_name
3. 와 같다면
    1. head의 type이 1이라면 head의 val이 반환값이다.
    2. head의 type이 2이라면
        line은head의 line이다.
        -1 반환값
4. head의exp_data==exp_name
5. 와 같은게 없다면 head는 head의 next를 해준다. head의 next가 NULL이 아닐때까지 반복해준준다.

getVal()


newnode.type = 1
newnode.exp_data = 'c'
newnode.val = 3
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'b'
newnode.val = 2
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'a'
newnode.val = 1
newnode.line = 0
next
Newnode
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
Newnode(
newnode.type = 2
newnode.exp_data = ' m'
newnode.val = 0
newnode.line = 13
next
Newnode
newnode.type = 2
newnode.exp_data = ' f'
newnode.val = 0
newnode.line = 6
next
Newnode
newnode.type = 2
newnode.exp_data = ' g'
newnode.val = 0
newnode.line = 1
next
Newnode
NULL
top
newnode.type = 1
newnode.exp_data = c
newnode.val = 2
newnode.line = 0
next
newnode.type = 1
newnode.exp_data = b
newnode.val = 1
newnode.line = 0
next
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
newnode.type = 1
newnode.exp_data = a
newnode.val = 3
newnode.line = 0
next
newnode.type = 2
newnode.exp_data = f
newnode.val = 0
newnode.line = 6
next
newnode.type = 3
newnode.exp_data = ' '
newnode.val = 0
newnode.line = 18
next
reset(f)
head

((b*c)+g(a));

```
//-1


////(3)-3-4-1
if ((retVal!=-1) & (retVal!=-999)){ ... }
////(3)-3-4-2
//-1
else
{

    ////(3)-3-4-2->1
    if (LastFunctionReturn==-999){ ... }
    ////(3)-3-4-2->2
    else

    {


    postfix[y]=LastFunctionReturn+48; /* in ascii table numeric values start from 48 */
    y++;
    i=i+3;
    LastFunctionReturn=-999;


    }
}
}

//1 증가
```

LastFunctionReturn= 3

postfix[3] =51

LastFunctionReturn=-999로 바꿔준다.

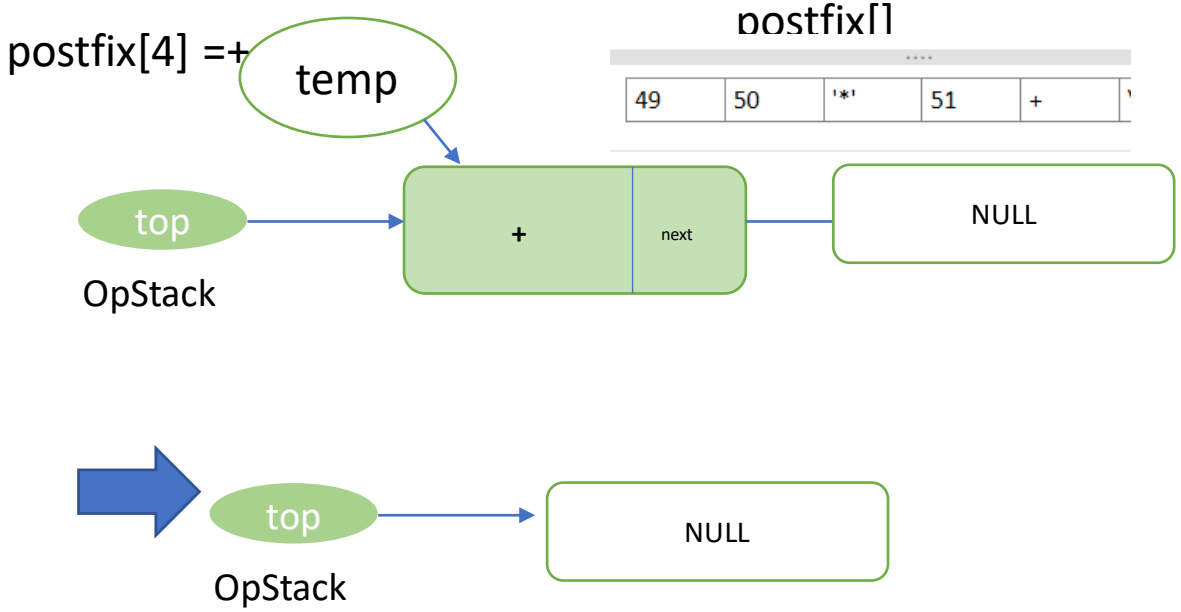
((b*c)+g(a));

```
//lineyedek[i]이 ')'이라면
else if (lineyedek[i]==')')
{
    //(3)-3-2-1
    ////0이면 true 1이면 false
    //0일때
    if (!isStackEmpty(MathStack) != 0 )
    {
        postfix[y]=PopOp(MathStack);
        y++; // 1증가
    }
}
```

①

②

postfix[4] =+



postfix[]

49	50	'*'	51	+

isStackEmpty

```
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)
1. 만약 stck의 top가 0이라면
   1. 반환값 1
2. 아니면 반환값 0

PopOp

```
char PopOp(OpStack *opstck)
{
opNode *temp;
char op;
if (opstck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
op=opstck->top->op;
temp=opstck->top;
opstck->top=opstck->top->next;
free(temp);
return op;
}
return NULL;
}
```

char PopOp(OpStack *opstck)
1. opNode구조체의 자료형 temp 포인터 선언
2. 문자자료형 op선언
3. 만약 opstck의 top이 NULL일 경우
   printf("ERROR, empty stack..."); 출력
4. 아니라면,
   1. op는 + 이다
   2. temp 는 opstck의 top이다.
   3. opstck의 top은 opstck의 top의 next이다.
   temp 동적메모리 해제
   반환값 op

반환값 NULL

```c
if (WillBreak==0)
                {
                /* get out items left in the mathstack */
                while (isStackEmpty(MathStack)==0)
                {

                    /* add the poped operator to the postfix */
                    postfix[y]=PopOp(MathStack);
                    y++;
                }

                postfix[y]='\0';

                //MathStack=FreeAll(MathStack);

                /* now calculate the postfix */
                /*printf("\nCURRENT POSTFIX=%s\n",postfix);*/

                i=0;

                CalcStack->top=NULL;
                while(postfix[i]!='\x0')
                {
                    if (isdigit(postfix[i])) {
                        /* push to stack */
                        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
                    }
                    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
                    {
                        val1=PopPostfix(CalcStack);

                        val2=PopPostfix(CalcStack);


                        switch (postfix[i])
                        {
                            case '+': resultVal=val2+val1;break;
                            case '-': resultVal=val2-val1;break;
                            case '/': resultVal=val2/val1;break;
                            case '*': resultVal=val2*val1;break;
                        }

                        CalcStack=PushPostfix(resultVal,CalcStack);
                    }
                    i++;
                }

                //CalcStack=FreeAll(CalcStack);
                LastExpReturn=CalcStack->top->val;


                }
```

①

isStackEmpty

```c
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)

1. 만약 stck의 top가 0이라면
    1. 반환값 1
2. 아니면 반환값 0

ⓐ
```
postfix[y]='\0';

//MathStack=FreeAll(MathStack);

/* now calculate the postfix */
/*printf("\nCURRENT POSTFIX=%s\n",postfix);*/

i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
        val1=PopPostfix(CalcStack);

        val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

Postfix[i]이 \0일때까지 반복실행

Postfix[i]가 숫자일시,

'0'은 아스키코드 48

1증가

Postfix[]

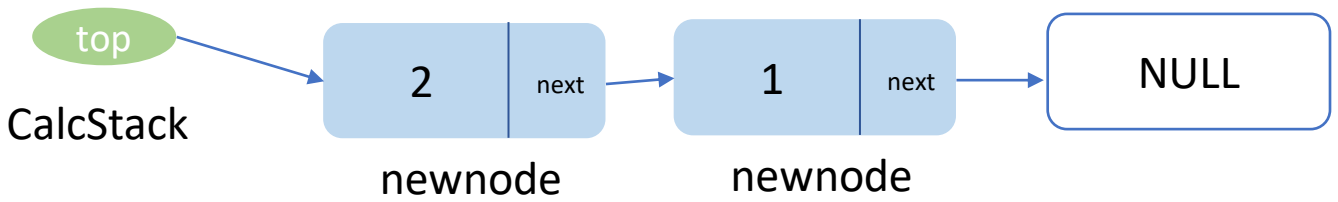
ⓐ

49	50	'*'	51	+	\0

```
PostfixStack * PushPostfix(int val,PostfixStack *poststck)
{
Postfixnode *newnode;
if ((newnode=(Postfixnode*)malloc(sizeof(Postfixnode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->val=val;
newnode->next=poststck->top;
poststck->top=newnode;
return poststck;
}
}
```

PostfixStack * PushPostfix(int val,PostfixStack *poststck)

1. Postfixnode구조체의 자료형인 newnode 포인터 선언
2. 만약 newnode는 Postfixnode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   NULL반환
3. 아니면라면,
   1. newnode의 val 은 매개변수 val이다
   2. newnode의 next 는 poststck의 top이다.
   3. poststck의 top 는 newnode이다.
   4. 반환값 poststck



```
i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
①       val1=PopPostfix(CalcStack);

②       val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

val1 = 2
val2=1

resultVal = 1*2

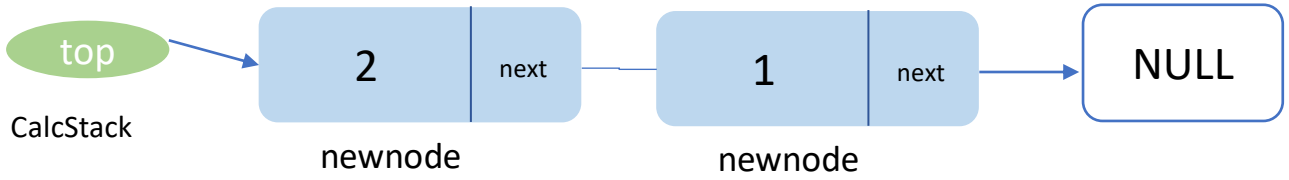
Postfix[]

49	50	'*'	51	+	\0

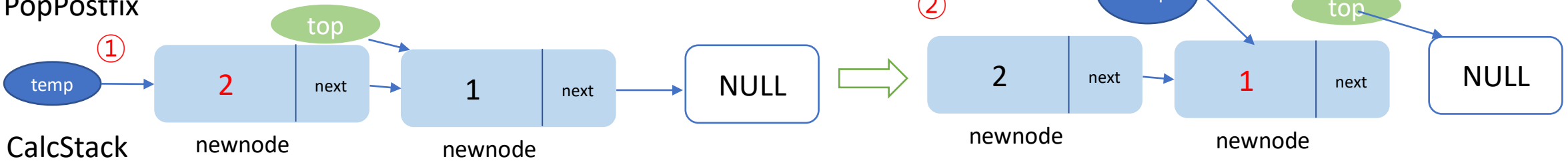
```
char PopPostfix(PostfixStack *poststck)
{
Postfixnode *temp;
int val;
if (poststck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
val=poststck->top->val;
temp=poststck->top;
poststck->top=poststck->top->next;
free(temp);
return val;
}
return NULL;
}
```

char PopPostfix(PstfixStack *poststcPostfixnode구조체의 자료k)

1. 형 temp 포인터 선언
2. 정수자료형인 val선언
3. 만약 poststck의 top이 NULL이라면
   1. ERROR, empty stack...console에 출력
4. 아니라면,
   1. val 은 poststck의 top의 val이다.
   2. tmep는 poststck의 top
   3. poststck의 top 은 poststck의 top의 next이다.
   4. temp의 동적메모리 해제
   5. 반환값 val
5. 반환값 NULL



PopPostfix



```
i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
        val1=PopPostfix(CalcStack);

        val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

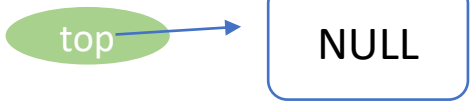
pushPostfix(2,CalcStck)

```
PostfixStack * PushPostfix(int val,PostfixStack *poststck)
{
Postfixnode *newnode;
if ((newnode=(Postfixnode*)malloc(sizeof(Postfixnode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->val=val;
newnode->next=poststck->top;
poststck->top=newnode;
return poststck;
}
}
```

PostfixStack * PushPostfix(int val,PostfixStack *poststck)

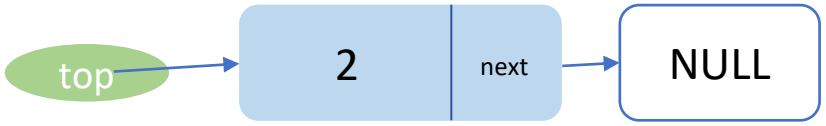
1. Postfixnode구조체의 자료형인 newnode 포인터 선언
2. 만약 newnode는 Postfixnode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   NULL반환
3. 아니면라면,
   1. newnode의 val 은 매개변수 val이다
   2. newnode의 next 는 poststck의 top이다.
   3. poststck의 top 는 newnode이다.
   4. 반환값 poststck

pushPostfix 전



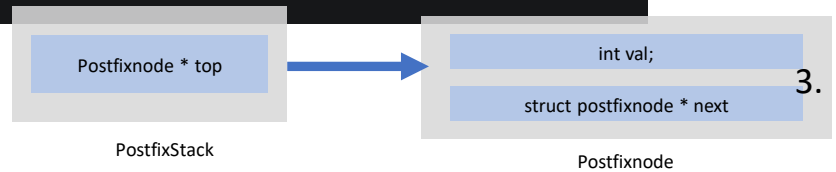
CalcStack

pushPostfix 후



CalcStack

newnode

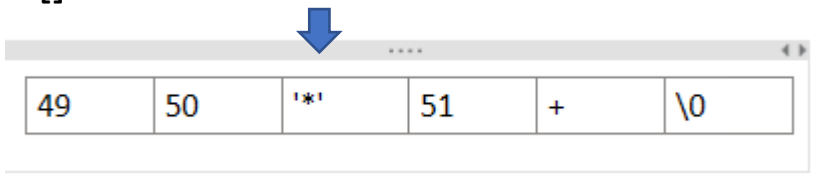


PostfixStack

Postfixnode

구조체 정리

Postfix[]



```
postfix[y]='\0';

//MathStack=FreeAll(MathStack);

/* now calculate the postfix */
/*printf("\nCURRENT POSTFIX=%s\n",postfix);*/

i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
        val1=PopPostfix(CalcStack);

        val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

Postfix[i]이 \0일때까지 반복실행

Postfix[i]가 숫자일시,

'0'은 아스키코드 48

1증가

Postfix[]

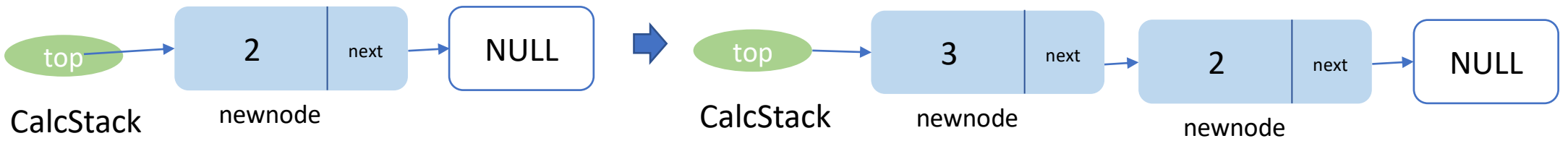
49	50	'*'	51	+	\0

```
PostfixStack * PushPostfix(int val,PostfixStack *poststck)
{
Postfixnode *newnode;
if ((newnode=(Postfixnode*)malloc(sizeof(Postfixnode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->val=val;
newnode->next=poststck->top;
poststck->top=newnode;
return poststck;
}
}
```

PostfixStack * PushPostfix(int val,PostfixStack *poststck)

1. Postfixnode구조체의 자료형인 newnode 포인터 선언
2. 만약 newnode는 Postfixnode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   NULL반환
3. 아니면라면,
   1. newnode의 val 은 3 이다
   2. newnode의 next 는 poststck의 top이다.
   3. poststck의 top 는 newnode이다.
   4. 반환값 poststck

PushPostfix()



```
i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
①   val1=PopPostfix(CalcStack);
②   val2=PopPostfix(CalcStack);

        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

val1 =3
val2=2

resultVal =2+3

pushPostfix(5,CalcStck)

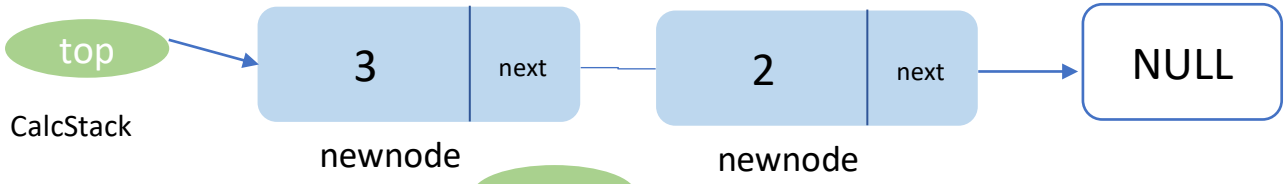
Postfix[]

49	50	'*'	51	+	\0

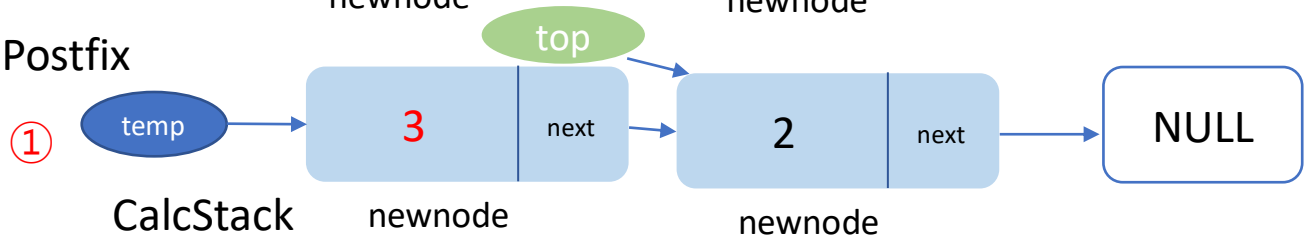
```
char PopPostfix(PostfixStack *poststck)
{
Postfixnode *temp;
int val;
if (poststck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
val=poststck->top->val;
temp=poststck->top;
poststck->top=poststck->top->next;
free(temp);
return val;
}
return NULL;
}
```

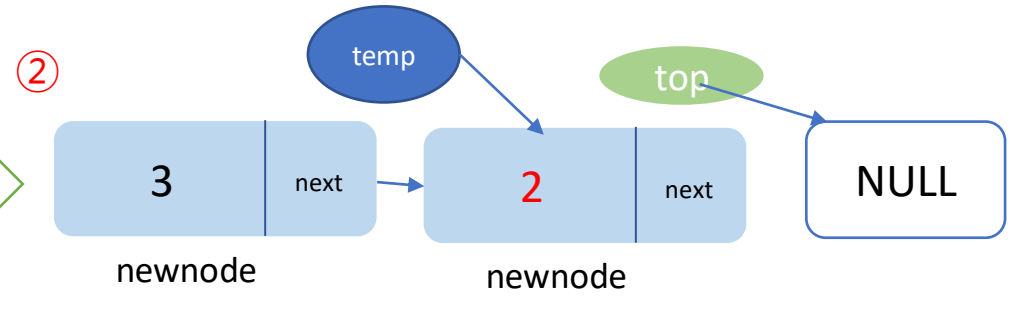
char PopPostfix(PstfixStack *poststcPostfixnode구조체의 자료k)

1. 형 temp 포인터 선언
2. 정수자료형인 val선언
3. 만약 poststck의 top이 NULL이라면
   1. ERROR, empty stack...console에 출력
4. 아니라면,
   1. val 은 poststck의 top의 val이다.
   2. tmep는 poststck의 top
   3. poststck의 top 은 poststck의 top의 next이다.
   4. temp의 동적메모리 해제
   5. 반환값 val
5. 반환값 NULL



PopPostfix



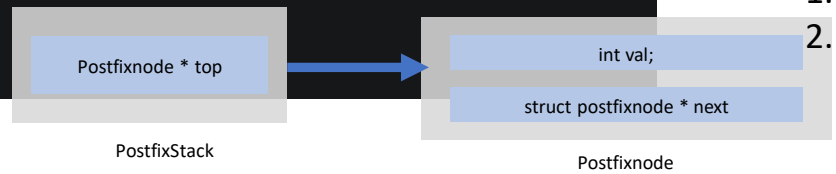


Postfix[]

49	50	'*'	51	+	\0

```
i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
        val1=PopPostfix(CalcStack);

        val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

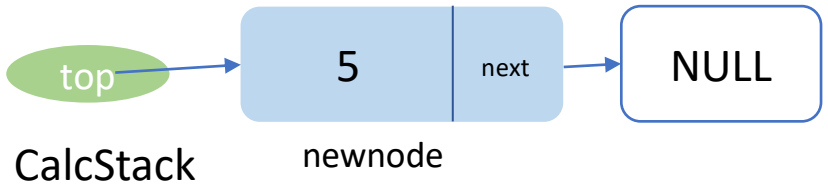
```
PostfixStack * PushPostfix(int val,PostfixStack *poststck)
{
Postfixnode *newnode;
if ((newnode=(Postfixnode*)malloc(sizeof(Postfixnode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->val=val;
newnode->next=poststck->top;
poststck->top=newnode;
return poststck;
}
}
```



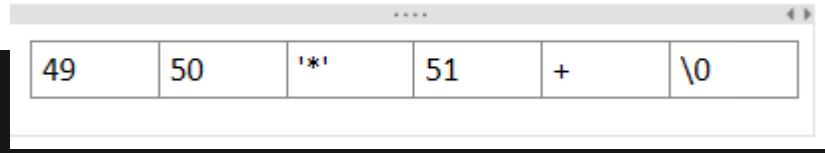
구조체 정리

PostfixStack * PushPostfix(int val,PostfixStack *poststck)

1. Postfixnode구조체의 자료형인 newnode 포인터 선언
2. 만약 newnode는 Postfixnode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   NULL반환
3. 아니면라면,
   1. newnode의 val 은 매개변수 val이다
   2. newnode의 next 는 poststck의 top이다.
   3. poststck의 top 는 newnode이다.
   4. 반환값 poststck



postfix[]

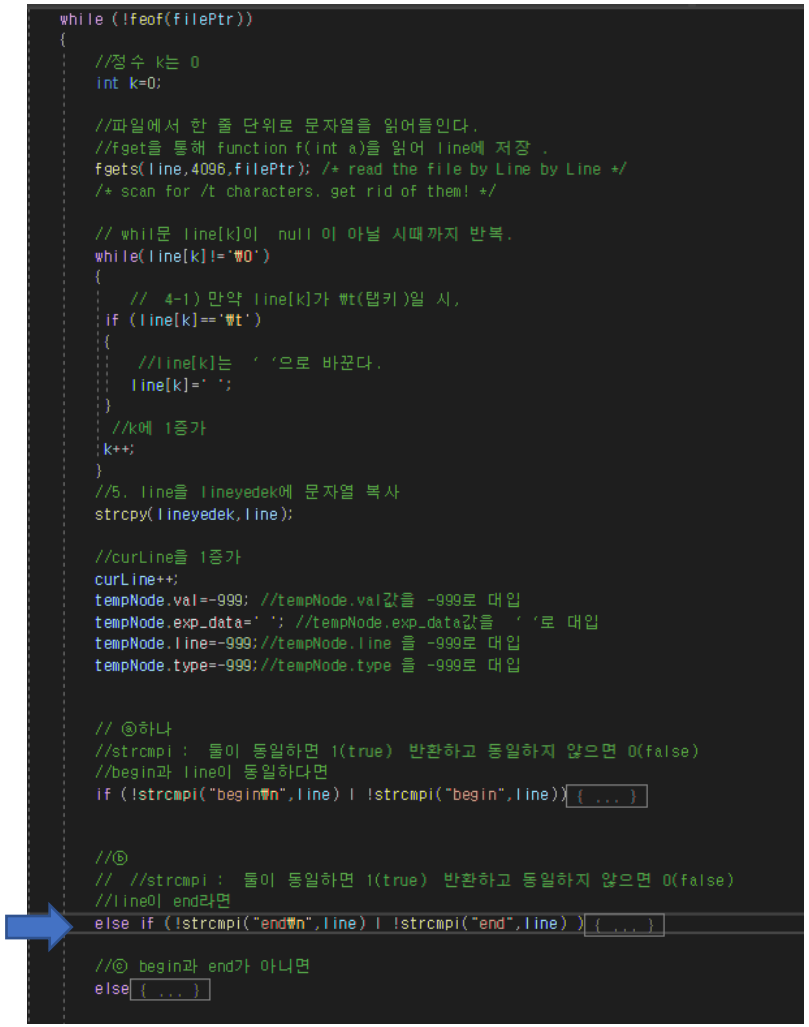


49	50	'*'	51	+	\0

```
i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
        val1=PopPostfix(CalcStack);

        val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

postfix[i] = \0이여서 while문 종료

LastExpReturn = 5

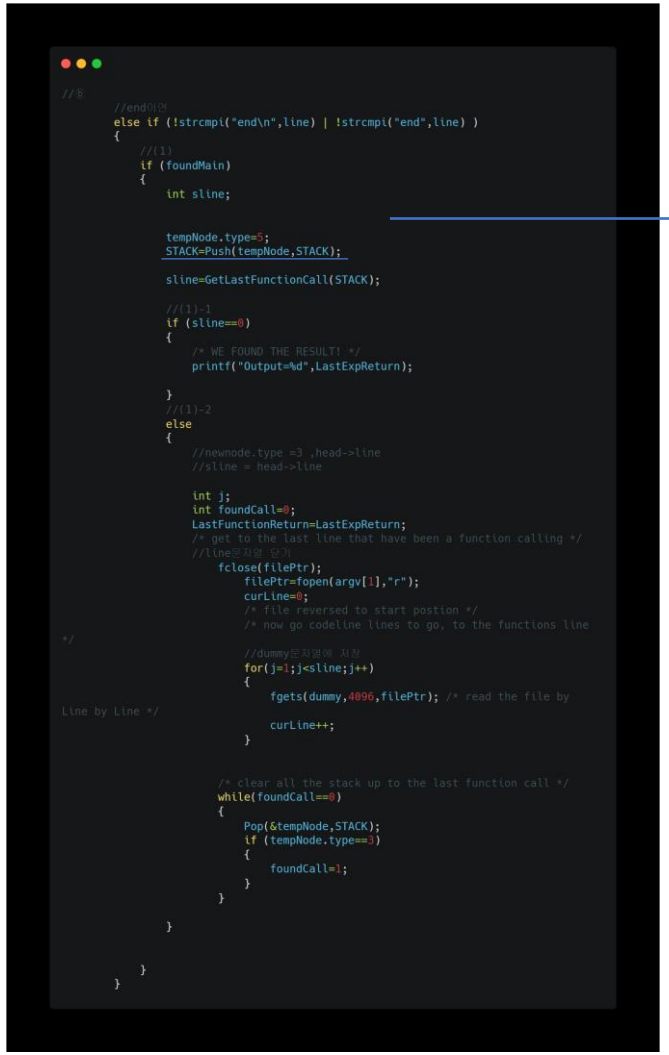


```
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //fget을 통해 function f(int a)을 읽어 line에 저장 .
    fgets(line,4096,filePtr); /* read the file by Line by Line */
    /* scan for /t characters. get rid of them! */

    // whil문 line[k]이  null 이 아닐 시때까지 반복.
    while(line[k]!='₩0')
    {
       //  4-1) 만약 line[k]가 ₩t(탭키)일 시,
     if (line[k]=='₩t')
     {
       //line[k]는  ' '으로 바꾼다.
      line[k]=' ';
     }
      //k에 1증가
     k++;
    }
    //5. line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을  ' '로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  둘이 동일하면 1(true) 반환하고 동일하지 않으면 0(false)
    //begin과 line이 동일하다면
    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }


    //ⓑ
    //  //strcmpi :  둘이 동일하면 1(true) 반환하고 동일하지 않으면 0(false)
    //line이 end라면
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }

    //ⓒ begin과 end가 아니면
    else{ ... }
```

input 2.sql (읽을 파일)

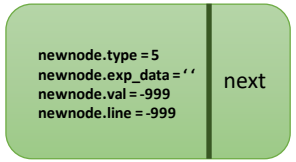
```
function g(int x)
begin
 (1+2-3+x);
end

function f(int a)
begin
  int b = 1;
  int c = 2;
  ((b*c)+g(a));
end

function main()
begin
  int a = 1;
  int b = 2;
  int c = 3;
  ((2 + f(c) ) * a);
end
```

line이 end일 때



```
//8
        //end이면
        else if (!strcmpi("end\n",line) | !strcmpi("end",line) )
        {
            //(1)
            if (foundMain)
            {
                int sline;


                tempNode.type=5;
                STACK=Push(tempNode,STACK);

                sline=GetLastFunctionCall(STACK);

                //(1)-1
                if (sline==0)
                {
                    /* WE FOUND THE RESULT! */
                    printf("Output=%d",LastExpReturn);

                }
                //(1)-2
                else
                {
                    //newnode.type =3 ,head->line
                    //sline = head->line

                    int j;
                    int foundCall=0;
                    LastFunctionReturn=LastExpReturn;
                    /* get to the last line that have been a function calling */
                    //line문자열 담기
                        fclose(filePtr);
                            filePtr=fopen(argv[1],"r");
                            curLine=0;
                            /* file reversed to start postion */
                            /* now go codeline lines to go, to the functions line
*/
                            //dummy문자열에 저장
                            for(j=1;j<sline;j++)
                            {
                                fgets(dummy,4096,filePtr); /* read the file by
Line by Line */
                                curLine++;
                            }


                            /* clear all the stack up to the last function call */
                        while(foundCall==0)
                        {
                            Pop(&tempNode,STACK);
                            if (tempNode.type==3)
                            {
                                foundCall=1;
                            }
                        }

                }


            }
        }
```

```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```



Stack * Push(Node sNode,Stack *stck)

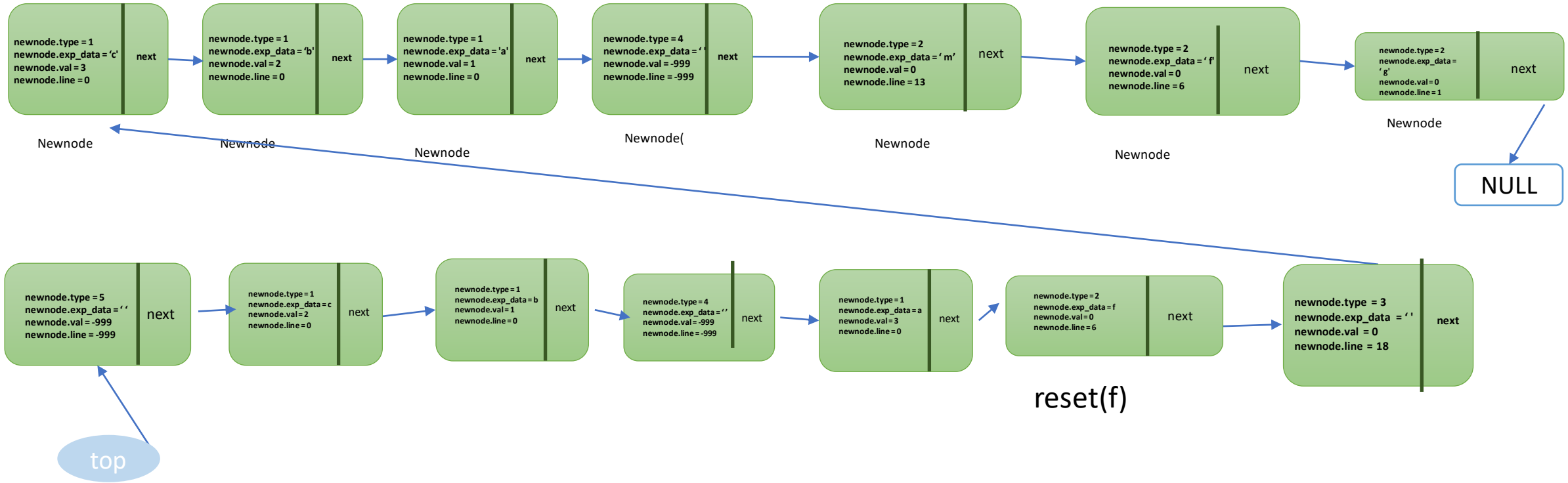
1. Node 자료형의 newnode포인터선언;
2. 만약 newnode는 Node타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   return NULL

3.아니라면

newnode의 type은 5의 타입이 된다.
newnode 의 val은 -999 이된다.
newnode 의 line 은 -999 이 된다.
newnode 의 next 는 stck의 top이 된다.
stck의 top은 newnode이다

반환값 stck

Push()


newnode.type = 1
newnode.exp_data = 'c'
newnode.val = 3
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'b'
newnode.val = 2
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'a'
newnode.val = 1
newnode.line = 0
next
Newnode
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
Newnode(
newnode.type = 2
newnode.exp_data = ' m'
newnode.val = 0
newnode.line = 13
next
Newnode
newnode.type = 2
newnode.exp_data = ' f'
newnode.val = 0
newnode.line = 6
next
Newnode
newnode.type = 2
newnode.exp_data = ' g'
newnode.val = 0
newnode.line = 1
next
Newnode
NULL
newnode.type = 5
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
top
newnode.type = 1
newnode.exp_data = c
newnode.val = 2
newnode.line = 0
next
newnode.type = 1
newnode.exp_data = b
newnode.val = 1
newnode.line = 0
next
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
newnode.type = 1
newnode.exp_data = a
newnode.val = 3
newnode.line = 0
next
newnode.type = 2
newnode.exp_data = f
newnode.val = 0
newnode.line = 6
next
reset(f)
newnode.type = 3
newnode.exp_data = ' '
newnode.val = 0
newnode.line = 18
next

line이 end일 때

sline=18

GetLastFunctionCall

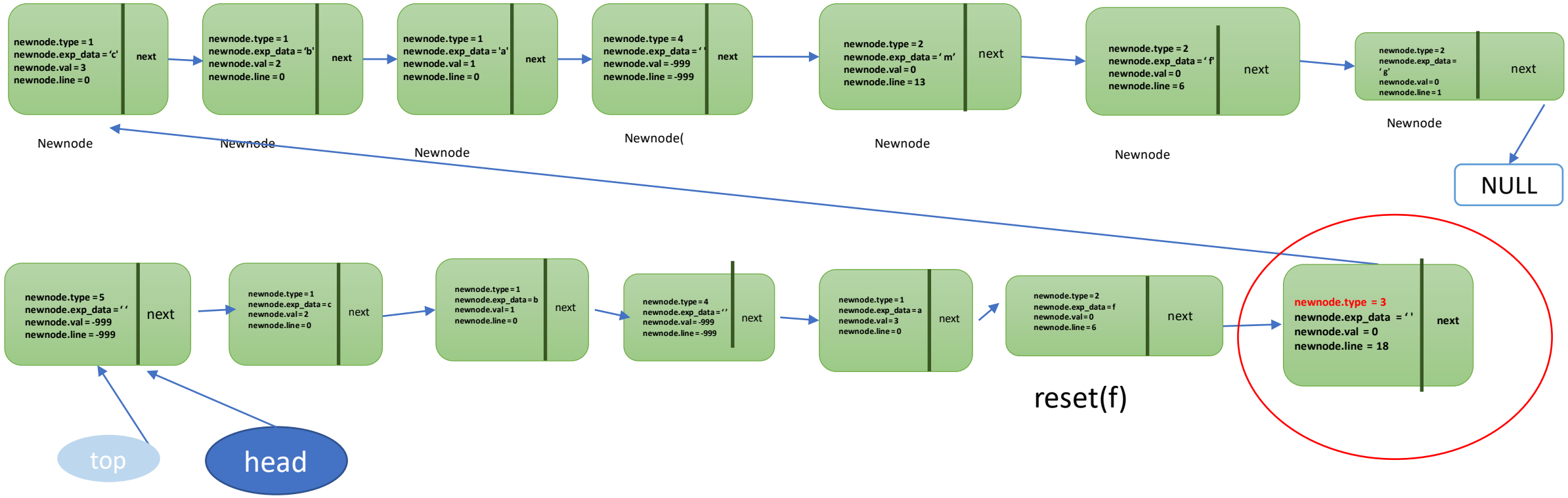
```
int GetLastFunctionCall(Stack *stck)
{
Node * head;

  if (stck->top == NULL )
  {
    printf("ERROR, empty stack...");
  }
  else
  {
    head=stck->top;
    do
    {
      if ( head->type==3 )
      {
        return head->line;
      }
      else
      {
        head=head->next;
      }
    } while (head->next!=NULL);
  }

  return 0;



}
```

18

int GetLastFunctionCall(Stack *stck)

1. Node의 구조체의 자료형인 포인터 head
2. 만약 stck의 top 이 NULL이라면,
    1. ERROR, empty stack… 이라고 콘솔 출력
3. 아니라면,
    1. head는 stck의 top이다.
    2. do
        1. head의 type은 3이라면
            1. 반환값 head의 line
        2. 아니라면
            1. head는 head의 next이다.
    3. while (head의 next가 NULL이 아니라면 반복)
4. 반환값 0

GetLastFunctionCall()



```
//(1)-2
else
{
    //newnode.type =3 ,head->line
    //sline = head->line

    int j;
    int foundCall=0;
    LastFunctionReturn=LastExpReturn;
    /* get to the last line that have been a function calling */
    //filePtr 닫기
        fclose(filePtr);
        /// filePtr= argv[1] 파일 읽기모드는 파일열기
            filePtr=fopen(argv[1],"r");
            curLine=0;
            /* file reversed to start postion */
            /* now go codeline lines to go, to the functions line */
            //dummy문자열에 저장
            for(j=1;j<sline;j++)
            {
                //
                fgets(dummy,4096,filePtr); /* read the file by Line by Line */
                curLine++;
            }


        /* clear all the stack up to the last function call */
        while(foundCall==0)
        {
            Pop(&tempNode,STACK);
            if (tempNode.type==3)
            {
                foundCall=1;
            }
        }

}
```

LastExpReturn = 5

LastFunctionReturn= 5

```
curLine=0;
sline=10
for문
//dummy문자열에  저장
for(j=1;j<18;j++)
{
//
fgets(dummy,4096,filePtr);
curLine++;//9
}
```

input 2.sql (읽을 파일)

```
function g(int x)
begin
 (1+2-3+x);
end

function f(int a)
begin
  int b = 1;
  int c = 2;
  ((b*c)+g(a));
end

function main()
begin
  int a = 1;
  int b = 2;
  int c = 3;
  ((2 + f(c) ) * a);
end
```

```
        }

    /* clear all the stack up to the last function call */
    while(foundCall==0)
    {
        Pop(&tempNode,STACK);
        if (tempNode.type==3)
        {
            foundCall=1;
        }
    }
}
```

tempNode.type=3이나오면
foundCall=1로 변경되면while문 종료

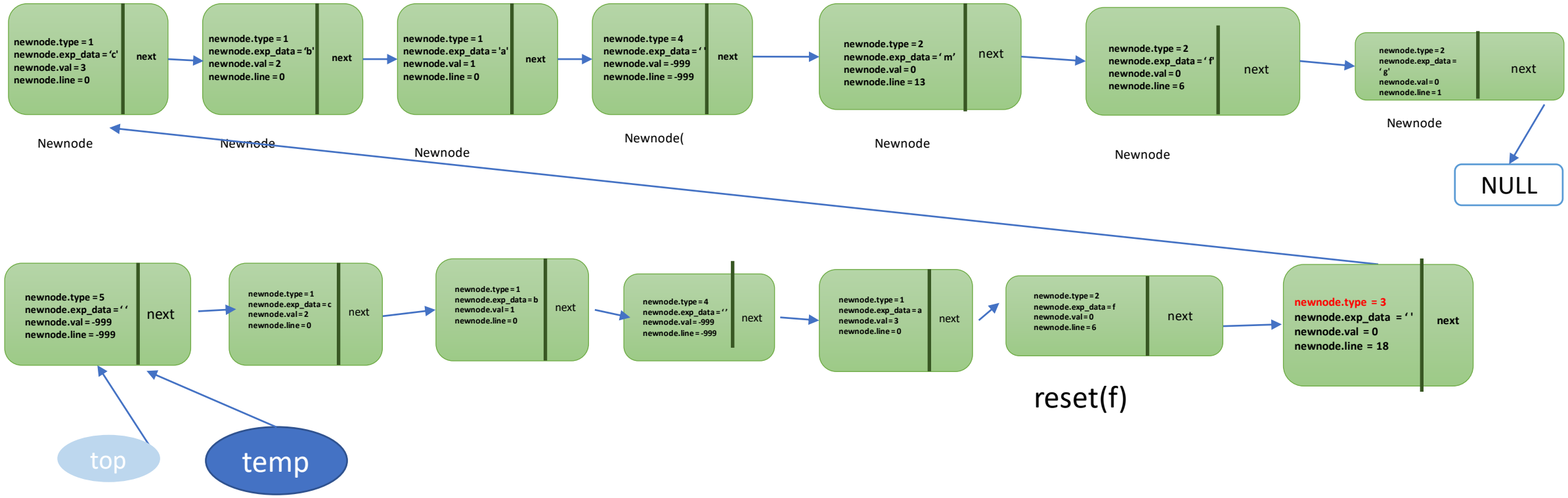
Pop

```
void Pop(Node * sNode,Stack *stck)
{
Node *temp;

if (stck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
sNode->exp_data=stck->top->exp_data;
sNode->type=stck->top->type;
sNode->line=stck->top->line;
sNode->val=stck->top->val;
temp=stck->top;
stck->top=stck->top->next;
free(temp);
}
}
```

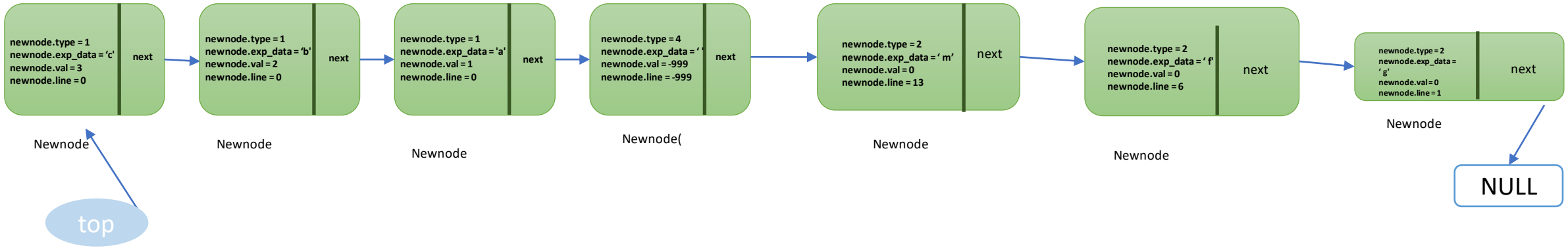
void Pop(Node * sNode,Stack *stck)

1. Node구조체 자료형인 temp 포인터 선언
2. 만약 stck의 top 이 NULL이라면,
    1. ERROR, empty stack... console출력
3. 아니라면
    1. sNode의 exp_data 는 stck의 top의 exp_data이다.
    2. sNode의 type는 stck의 top의 type 이다
    3. sNode의 line는 stck의 top의 line이다
    4. sNode의 val는 stck의 top의 val이다
    5. temp는 stck의 top이다.
    6. stck의 top은 stck의 top의 next이다.
    7. temp동적메모리 해제

Pop() 전


newnode.type = 1
newnode.exp_data = 'c'
newnode.val = 3
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'b'
newnode.val = 2
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'a'
newnode.val = 1
newnode.line = 0
next
Newnode
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
Newnode(
newnode.type = 2
newnode.exp_data = ' m'
newnode.val = 0
newnode.line = 13
next
Newnode
newnode.type = 2
newnode.exp_data = ' f'
newnode.val = 0
newnode.line = 6
next
Newnode
newnode.type = 2
newnode.exp_data = ' g'
newnode.val = 0
newnode.line = 1
next
Newnode
NULL
newnode.type = 5
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
newnode.type = 1
newnode.exp_data = c
newnode.val = 2
newnode.line = 0
next
newnode.type = 1
newnode.exp_data = b
newnode.val = 1
newnode.line = 0
next
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
newnode.type = 1
newnode.exp_data = a
newnode.val = 3
newnode.line = 0
next
newnode.type = 2
newnode.exp_data = f
newnode.val = 0
newnode.line = 6
next
reset(f)
newnode.type = 3
newnode.exp_data = ' '
newnode.val = 0
newnode.line = 18
next
top
temp

Pop() 후


newnode.type = 1
newnode.exp_data = 'c'
newnode.val = 3
newnode.line = 0
next
newnode.type = 1
newnode.exp_data = 'b'
newnode.val = 2
newnode.line = 0
next
newnode.type = 1
newnode.exp_data = 'a'
newnode.val = 1
newnode.line = 0
next
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
newnode.type = 2
newnode.exp_data = ' m'
newnode.val = 0
newnode.line = 13
next
newnode.type = 2
newnode.exp_data = ' f'
newnode.val = 0
newnode.line = 6
next
newnode.type = 2
newnode.exp_data = ' g'
newnode.val = 0
newnode.line = 1
next
Newnode
Newnode
Newnode
Newnode(
Newnode
Newnode
Newnode
top
NULL

((2 + f(c)) * a);

```
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //fget을 통해 function f(int a)을 읽어 line에 저장 .
    fgets(line,4096,filePtr); /* read the file by Line by Line */
    /* scan for /t characters. get rid of them! */

    // whil문 line[k]이  null 이 아닐 시때까지 반복.
    while(line[k]!='₩0')
    {
        //  4-1) 만약 line[k]가 ₩t(탭키)일 시,
     if (line[k]=='₩t')
     {
        //line[k]는  ' '으로 바꾼다.
       line[k]=' ';
     }
      //k에 1증가
     k++;
    }
    //5. line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을  ' '로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  들이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //begin과 line이 동일하다면
    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }


    //ⓑ
    //  //strcmpi :  들이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //line이 end라면
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }

    //ⓒ begin과 end가 아니면
    else{ ... }
```

input 2.sql (읽을 파일)

```
function g(int x)
begin
  (1+2-3+x);
end

function f(int a)
begin
   int b = 1;
   int c = 2;
   ((b*c)+g(a));
end

function main()
begin
   int a = 1;
   int b = 2;
   int c = 3;
   ((2 + f(c) ) * a);
end
```

((2 + f(c)) * a);

firstword[0]= (라면

```
else if (firstword[0]=='(')
{

//foundMain == 0이면 false 0이 아니면 True
if (foundMain)
   {

   int i=0;
   int y=0; //3

   //OpStack + MathStack
   //MathStack.top은 NULL
   MathStack->top=NULL;
   /* now make the postfix calculation */



   //(3)-2
   //lineyedek[i]이 NULL이 아닐때까지 반복
   while(lineyedek[i]!='₩x0')
   {
      //(3)-3-1
      /* evulate the function */
      //숫자라면 true
      if (isdigit(lineyedek[i])){ ... }
       /* ... */

      //(3)-3-2
      //lineyedek[i]이 ')'이라면
      else if (lineyedek[i]==')'){ ... }

      ////(3)-3-3
      //lineyedek[i]이 '+'이거나 lineyedek[i]이 '-' 이거나 lineyedek[i]이 '+' 이거나 lineyedek[i]이 '/'이라면
      else if ((lineyedek[i]=='+') | (lineyedek[i]=='-') | (lineyedek[i]=='+') | (lineyedek[i]=='/')){ ... }

      ////(3)-3-4
      //알파벳 대문자 "A-Z"는 1를 반환,알파벳 소문자 'a-z"는 2를 반환.
      //lineyedek[i] 영어라면
      else if (isalpha(lineyedek[i])>0){ ... }

      //1 증가
      i++;
   }//while문
```

```
/* evulate the function */
//숫자라면 true
if (isdigit(lineyedek[i])) {
   postfix[y]=lineyedek[i];
   y++;

}
 /*      */
```

postfix[0] = 2

(2 + f(c)) * a);

만약 isStackEmpty( ) 0이아니라면

```
////(3)-3-3
//lineyedek[i]이 '+'이거나 lineyedek[i]이 '-' 이거나 lineyedek[i]이 '+' 이거나 lineyedek[i]이 '/'이라면
else if ((lineyedek[i]=='+') | (lineyedek[i]=='-') | (lineyedek[i]=='*') | (lineyedek[i]=='/'))
{
   ////(3)-3-3-1
   /*operators*/
   //00이면 false 10이면 true
   //처음일 때
   if (isStackEmpty(MathStack) != 0 ) //1일때 - stck->top==0
   {
/* if stack empty push the operator to stack */
      //+
      MathStack=PushOp(lineyedek[i],MathStack);
   }
   ////(3)-3-3-2

   //0일때 - stck -top = 00이 아닐때
   //처음이 아닐때
   else { ... }
}
```

ⓐ

ⓑ

PushOp(+, MathStack)

isStackEmpty

```
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)

1. 만약 stck의 top가 0이라면
   1. 반환값 1
2. 아니면 반환값 0

PushOp

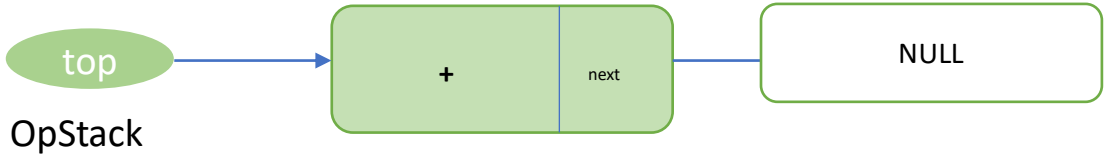
```
OpStack * PushOp(char op,OpStack *opstck)
{
opNode *newnode;
if ((newnode=(opNode*)malloc(sizeof(opNode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->op=op;
newnode->next=opstck->top;
opstck->top=newnode;
return opstck;
}
}
```

OpStack * PushOp(char op,OpStack *opstck)

1. opNode자료형의 newnode포인터선언;
2. 만약 newnode는 opNode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   반환NULL
3. 아니라면,
   1. newnode의 op는 매개변수+이다.
   2. newnode의 next는 opstck의 top이다.
   3. opstck의 top은 newnode이다.

반환값 opstck

PushOp



(2 + f(c)) * a);

```
//lineyedek[i] 영어라면
else if (isalpha(lineyedek[i])>0)
{
    int codeline=0;
    int dummyint=0;
    /*look if it's a variable or function call
    int retVal=0;
    retVal=GetVal(lineyedek[i],&codeline,STACK);
    //int codeline =1

    //-1


    ////(3)-3-4-1
    if ((retVal!=-1) & (retVal!=-999)){ ... }
    ////(3)-3-4-2
    //-1
    else{ ... }
}
```

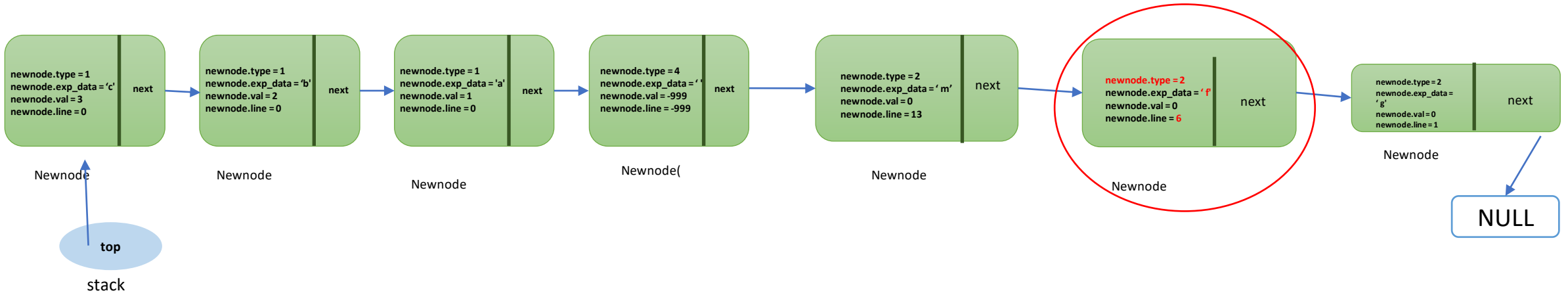
GetVal (f,&codeline,STACK)

GetVal

```
int GetVal(char exp_name,int * line,Stack *stck)
{

Node * head;
*line=0;
    if (stck->top == NULL )
    {
        printf("ERROR, empty stack...");
    }
    else
    {
        head=stck->top;
        do
        {
            if ( head->exp_data==exp_name )
            {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }
            }
            else
            {
                head=head->next;
            }
        } while (head->next!=NULL);
        /* check agin once more */
        if (head->exp_data==exp_name)
        {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }


        }
    }

    return -999;
}
```

head->exp_data = f

*line = 6
return -1



(2 + f(c)) * a);

LastFunctionReturn= 5

```
//-1


////(3)-3-4-1
if ((retVal!=-1) & (retVal!=-999)){ ... }
////(3)-3-4-2
//-1
else
{

    ////(3)-3-4-2->1
    if (LastFunctionReturn==-999){ ... }
    ////(3)-3-4-2->2
    else

    {


    postfix[y]=LastFunctionReturn+48; /* in ascii table numeric values start from 48 */
    y++;
    i=i+3;
    LastFunctionReturn=-999;


    }
}
}

//1 증가
```

postfix[1] =53

postfix[]

2	53

LastFunctionReturn=-999변한다.

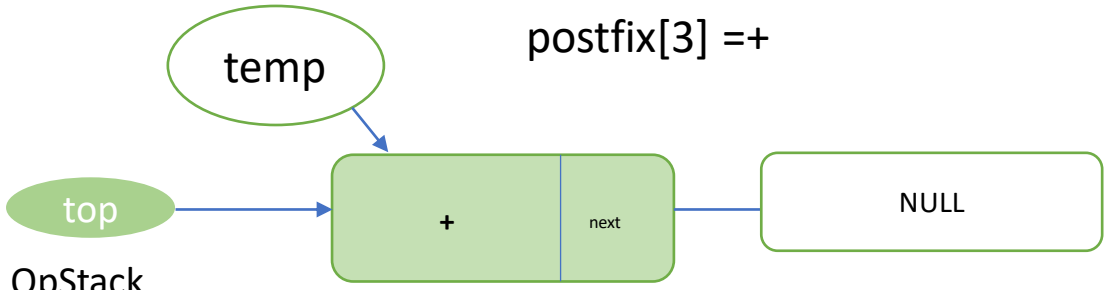
(2 + f(c)) * a);

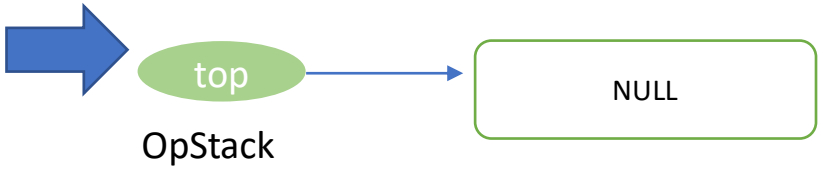
```
//lineyedek[i]이 ')'이라면
else if (lineyedek[i]==')')
{
    //(3)-3-2-1
    ////0이면 true 1이면 false
    //0일때
    if (!isStackEmpty(MathStack) != 0 )
    {
        postfix[y]=PopOp(MathStack);
        y++; // 1증가
    }
}
```

①

②

postfix[3] =+





isStackEmpty

```
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)
1. 만약 stck의 top가 0이라면
   1. 반환값 1
2. 아니면 반환값 0

PopOp

```
char PopOp(OpStack *opstck)
{
opNode *temp;
char op;
if (opstck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
op=opstck->top->op;
temp=opstck->top;
opstck->top=opstck->top->next;
free(temp);
return op;
}
return NULL;
}
```

char PopOp(OpStack *opstck)
1. opNode구조체의 자료형 temp 포인터 선언
2. 문자자료형 op선언
3. 만약 opstck의 top이 NULL일 경우
   printf("ERROR, empty stack..."); 출력

4. 아니라면,
   1. op는 + 이다
   2. temp 는 opstck의 top이다.
   3. opstck의 top은 opstck의 top의 next이다.
   temp 동적메모리 해제
   반환값 op

반환값 NULL

(2 + f(c)) * a);

```
////(3)-3-3
//lineyedek[i]이 '+'이거나 lineyedek[i]이 '-' 이거나 lineyedek[i]이 '+' 이거나 lineyedek[i]이 '/'이라면
else if ((lineyedek[i]=='+') | (lineyedek[i]=='-') | (lineyedek[i]=='*') | (lineyedek[i]=='/'))
{
    ////(3)-3-3-1
    /*operators*/
    //00이면 false 10이면 true
    //처음일 때
    if (isStackEmpty(MathStack) != 0 ) //- stck->top==0
    {
/* if stack empty push the operator to stack */
      //+
      MathStack=PushOp(lineyedek[i],MathStack);
    }
    ////(3)-3-3-2
```

PushOp(*,MathStack)

isStackEmpty

```
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)

1. 만약 stck의 top가 0이라면
   1. 반환값 1
2. 아니면 반환값 0

PushOp

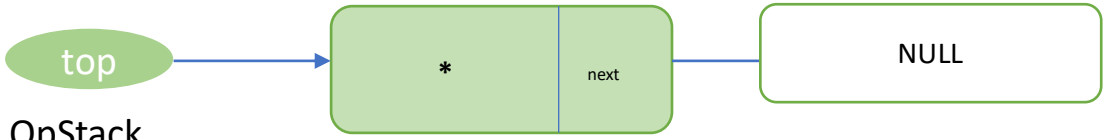
```
OpStack * PushOp(char op,OpStack *opstck)
{
opNode *newnode;
if ((newnode=(opNode*)malloc(sizeof(opNode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->op=op;
newnode->next=opstck->top;
opstck->top=newnode;
return opstck;
}
```

OpStack * PushOp(char op,OpStack *opstck)

1. opNode자료형의 newnode포인터선언;
2. 만약 newnode는 opNode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   반환NULL
3. 아니라면,
   1. newnode의 op는 매개변수op이다.
   2. newnode의 next는 opstck의 top이다.
   3. opstck의 top은 newnode이다.

반환값 opstck

PushOp



OpStack

MathStack

(2 + f(c)) * a);

```
//lineyedek[i] 영어라면
else if (isalpha(lineyedek[i])>0)
{
    int codeline=0;
    int dummyint=0;
    /*look if it's a variable or function call
    int retVal=0;
    retVal=GetVal(lineyedek[i],&codeline,STACK);
    //int codeline =1

    //-1


    ////(3)-3-4-1
    if ((retVal!=-1) & (retVal!=-999)){ ... }
    ////(3)-3-4-2
    //-1
    else{ ... }
}
```

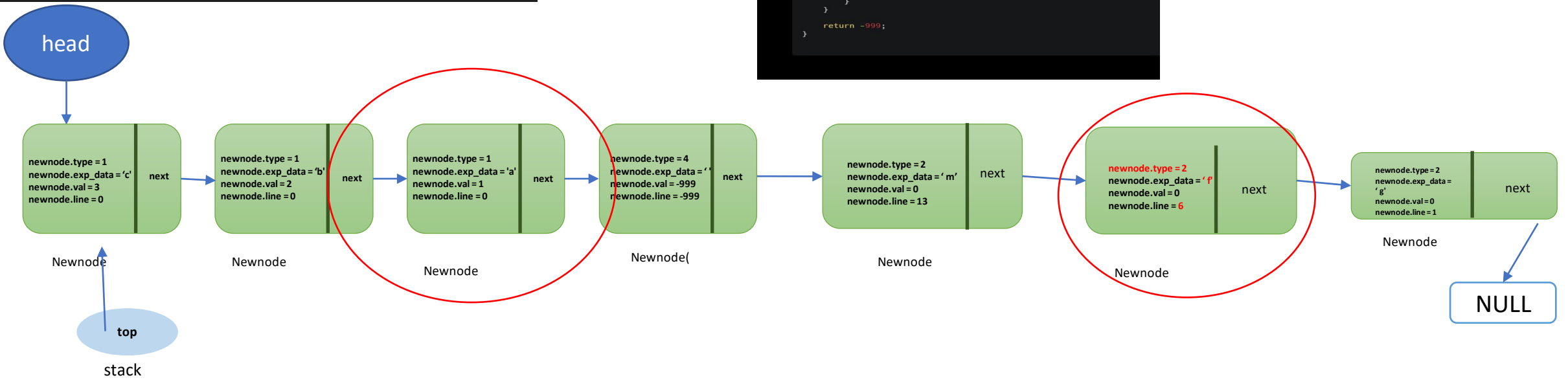
GetVal (a,&codeline,STACK)

GetVal

```
int GetVal(char exp_name,int * line,Stack *stck)
{

Node * head;
*line=0;
    if (stck->top == NULL )
    {
        printf("ERROR, empty stack...");
    }
    else
    {
        head=stck->top;
        do
        {
            if ( head->exp_data==exp_name )
            {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }

            }
            else
            {
                head=head->next;
            }
        } while (head->next!=NULL);
        /* check agin once more */
        if (head->exp_data==exp_name)
        {

                if (head->type==1)
                {
                    /* return the variables value */
                    return head->val;
                }
                else if (head->type==2)
                {
                    *line=head->line;
                    return -1;
                    /* it's a function so return -1 */
                }


        }
    }

    return -999;
}
```

head->exp_data = a

return 1



(2 + f(c)) * a);

postfix[4] =49

```
////(3)-3-4-1
if ((retVal!=-1) & (retVal!=-999))
{
    /* if variable */
    postfix[y]=retVal+48; /* in as
    y++;

}
```

(2 + f(c)) * a);

```
//(3)-3-2
//lineyedek[i]이 ')'이라면
else if (lineyedek[i]==')')
{
    //(3)-3-2-1
    ////00이면 true 10이면 false
    //0일때
    if (!isStackEmpty(MathStack) != 0 )
    {
        postfix[y]=PopOp(MathStack);
        y++; // 1증가
    }
}
```

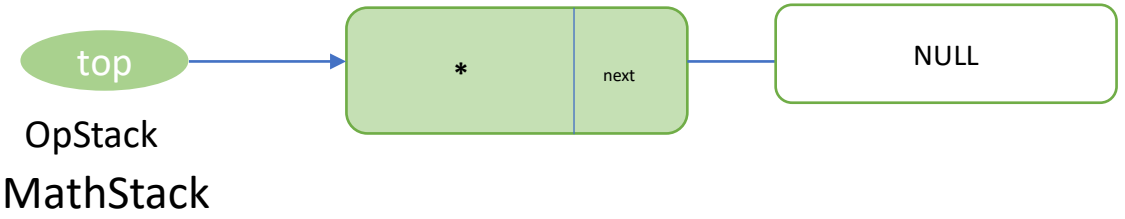
isStackEmpty

```
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)

1. 만약 stck의 top가 0이라면
    1. 반환값 1
2. 아니면 반환값 0

postfix[]



(2 + f(c)) * a);

```
//(3)-3-2
//lineyedek[i]이 ')'이라면
else if (lineyedek[i]==')')
{
    //(3)-3-2-1
    ////00이면 true 1이면 false
    //0일때
    if (!isStackEmpty(MathStack) != 0 )
    {
        postfix[y]=PopOp(MathStack);
        y++; // 1증가
    }
}
```

isStackEmpty

```
int isStackEmpty(OpStack *stck)
{
  if (stck->top==0)
    return 1;
return 0;
}
```

int isStackEmpty(OpStack *stck)

1. 만약 stck의 top가 0이라면
   1. 반환값 1
2. 아니면 반환값 0

PopOp

```
char PopOp(OpStack *opstck)
{
opNode *temp;
char op;
if (opstck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
op=opstck->top->op;
temp=opstck->top;
opstck->top=opstck->top->next;
free(temp);
return op;
}
return NULL;
}
```

char PopOp(OpStack *opstck)

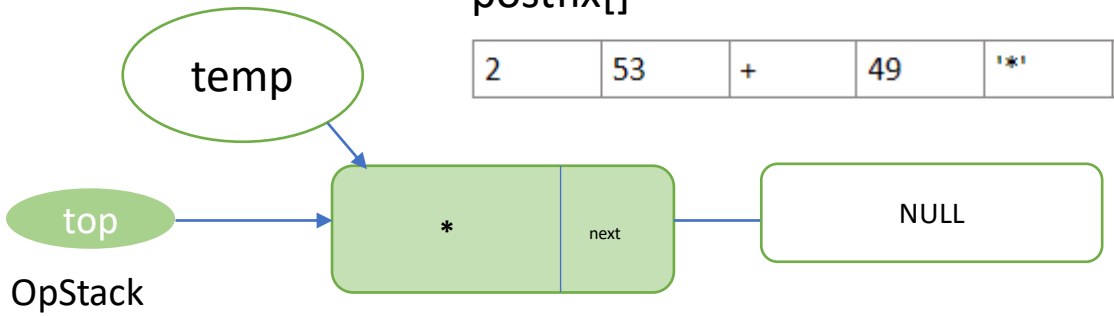
1. opNode구조체의 자료형 temp 포인터 선언
2. 문자자료형 op선언
3. 만약 opstck의 top이 NULL일 경우
   printf("ERROR, empty stack..."); 출력
4. 아니라면;
   1. op는 * 이다
   2. temp 는 opstck의 top이다.
   3. opstck의 top은 opstck의 top의 next이다.
   temp 동적메모리 해제
   반환값 op

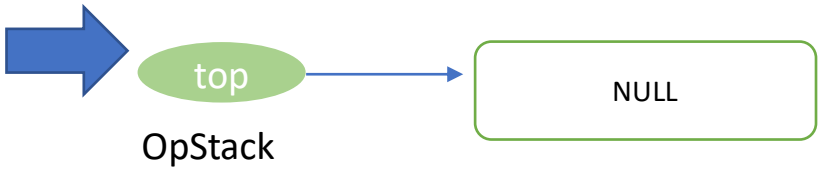
반환값 NULL

postfix[5] =*

postfix[]

2	53	+	49	'*'	\0





MathStack

ⓐ

```
postfix[y]='\0';

//MathStack=FreeAll(MathStack);

/* now calculate the postfix */
/*printf("\nCURRENT POSTFIX=%s\n",postfix);*/

i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
        val1=PopPostfix(CalcStack);

        val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

Postfix[i]이 \0일때까지 반복실행

Postfix[i]가 숫자일시,

'0'은 아스키코드 48

1증가

Postfix[]

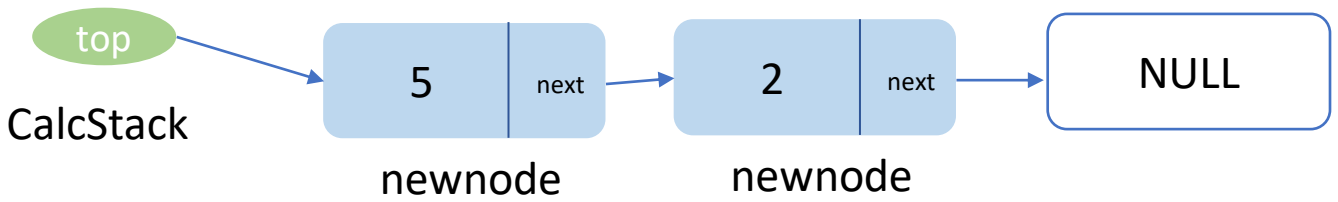
ⓐ

2	53	+	49	'*'	\0

```
PostfixStack * PushPostfix(int val,PostfixStack *poststck)
{
Postfixnode *newnode;
if ((newnode=(Postfixnode*)malloc(sizeof(Postfixnode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->val=val;
newnode->next=poststck->top;
poststck->top=newnode;
return poststck;
}
}
```

PostfixStack * PushPostfix(int val,PostfixStack *poststck)

1. Postfixnode구조체의 자료형인 newnode 포인터 선언
2. 만약 newnode는 Postfixnode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   NULL반환
3. 아니면라면,
   1. newnode의 val 은 매개변수 val이다
   2. newnode의 next 는 poststck의 top이다.
   3. poststck의 top 는 newnode이다.
   4. 반환값 poststck



```
i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
①       val1=PopPostfix(CalcStack);

②       val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

val1 = 5
val2=2

resultVal = 2+5

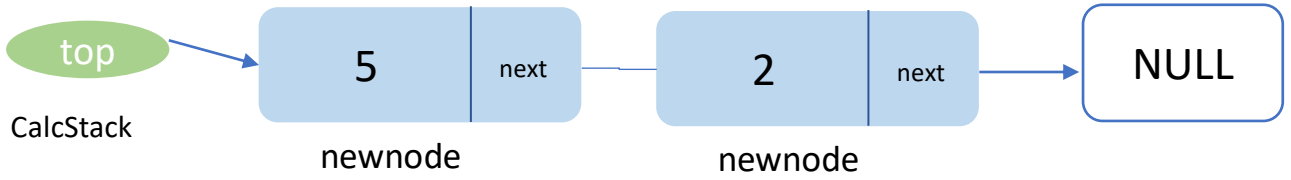
Postfix[]

2	53	+	49	'*'	\0

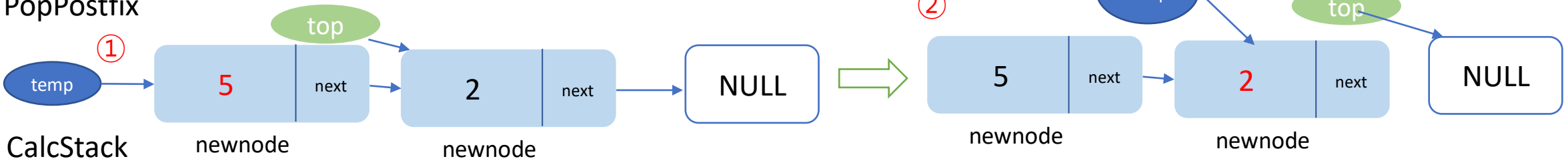
```
char PopPostfix(PostfixStack *poststck)
{
Postfixnode *temp;
int val;
if (poststck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
val=poststck->top->val;
temp=poststck->top;
poststck->top=poststck->top->next;
free(temp);
return val;
}
return NULL;
}
```

char PopPostfix(PstfixStack *poststcPostfixnode구조체의 자료k)

1. 형 temp 포인터 선언
2. 정수자료형인 val선언
3. 만약 poststck의 top이 NULL이라면
   1. ERROR, empty stack...console에 출력
4. 아니라면,
   1. val 은 poststck의 top의 val이다.
   2. tmep는 poststck의 top
   3. poststck의 top 은 poststck의 top의 next이다.
   4. temp의 동적메모리 해제
   5. 반환값 val
5. 반환값 NULL



PopPostfix



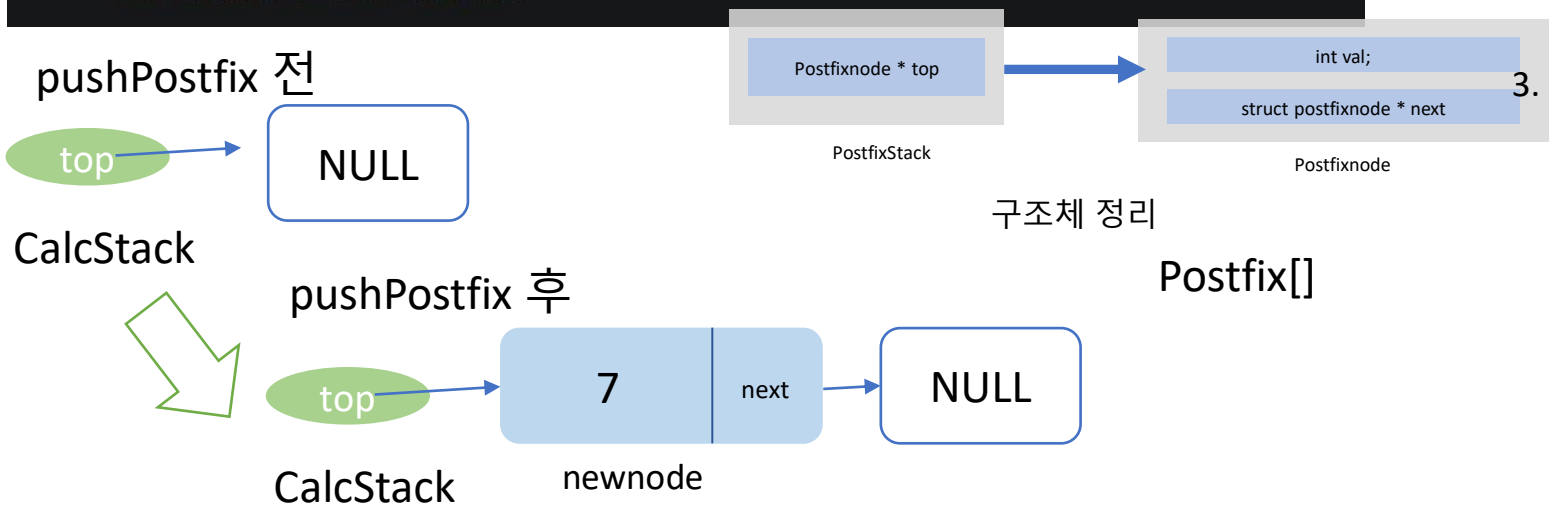
```
i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
        val1=PopPostfix(CalcStack);

        val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

pushPostfix(7,CalcStck)

```
PostfixStack * PushPostfix(int val,PostfixStack *poststck)
{
Postfixnode *newnode;
if ((newnode=(Postfixnode*)malloc(sizeof(Postfixnode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->val=val;
newnode->next=poststck->top;
poststck->top=newnode;
return poststck;
}
}
```

PostfixStack * PushPostfix(int val,PostfixStack *poststck)

1. Postfixnode구조체의 자료형인 newnode 포인터 선언
2. 만약 newnode는 Postfixnode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   NULL반환
3. 아니면라면,
   1. newnode의 val 은 매개변수 val이다
   2. newnode의 next 는 poststck의 top이다.
   3. poststck의 top 는 newnode이다.
   4. 반환값 poststck



구조체 정리

Postfix[]

```
postfix[y]='\0';

//MathStack=FreeAll(MathStack);

/* now calculate the postfix */
/*printf("\nCURRENT POSTFIX=%s\n",postfix);*/

i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
        val1=PopPostfix(CalcStack);

        val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

Postfix[i]이 \0일때까지 반복실행

Postfix[i]가 숫자일시,

'0'은 아스키코드 48

1증가

Postfix[]

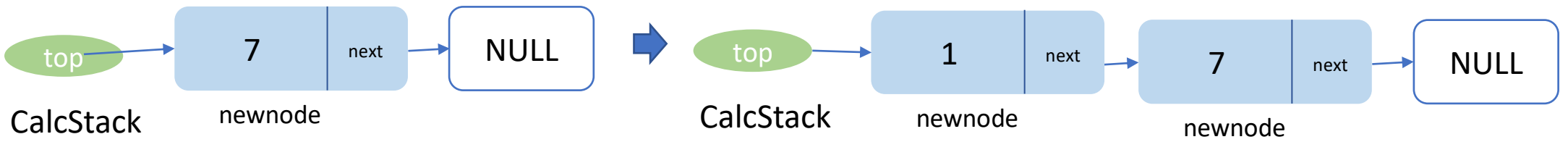
2	53	+	49	'*'	\0

```
PostfixStack * PushPostfix(int val,PostfixStack *poststck)
{
Postfixnode *newnode;
if ((newnode=(Postfixnode*)malloc(sizeof(Postfixnode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->val=val;
newnode->next=poststck->top;
poststck->top=newnode;
return poststck;
}
}
```

PostfixStack * PushPostfix(int val,PostfixStack *poststck)

1. Postfixnode구조체의 자료형인 newnode 포인터 선언
2. 만약 newnode는 Postfixnode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   NULL반환
3. 아니면라면,
   1. newnode의 val 은 1 이다
   2. newnode의 next 는 poststck의 top이다.
   3. poststck의 top 는 newnode이다.
   4. 반환값 poststck

PushPostfix()



```
i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
①   val1=PopPostfix(CalcStack);

②   val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

val1 = 1
val2=7

resultVal =7*1

pushPostfix(7,CalcStck)

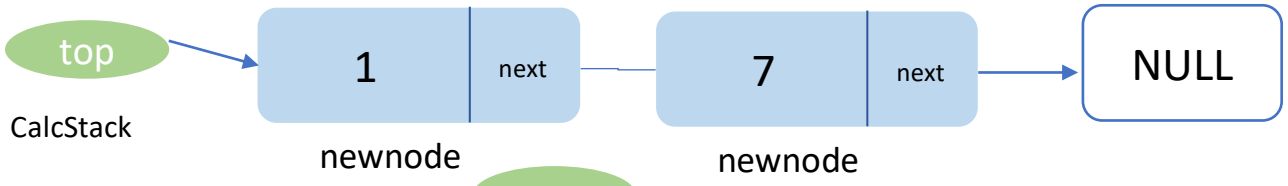
Postfix[]

2	51	+	49	'*'	\0

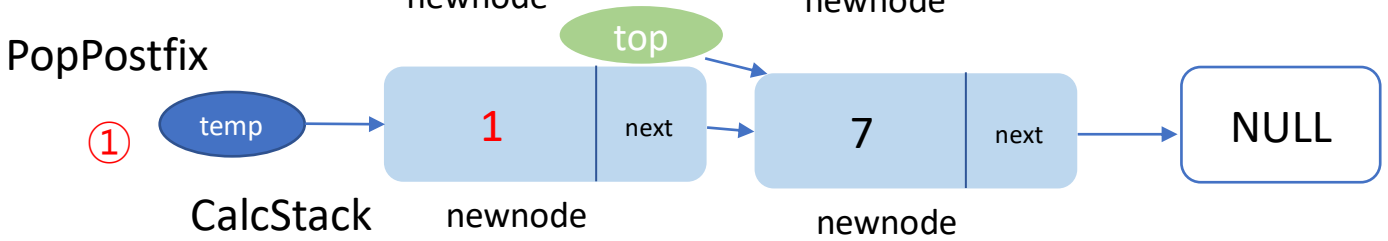
```
char PopPostfix(PostfixStack *poststck)
{
Postfixnode *temp;
int val;
if (poststck->top == NULL )
{
printf("ERROR, empty stack...");
}
else
{
val=poststck->top->val;
temp=poststck->top;
poststck->top=poststck->top->next;
free(temp);
return val;
}
return NULL;
}
```

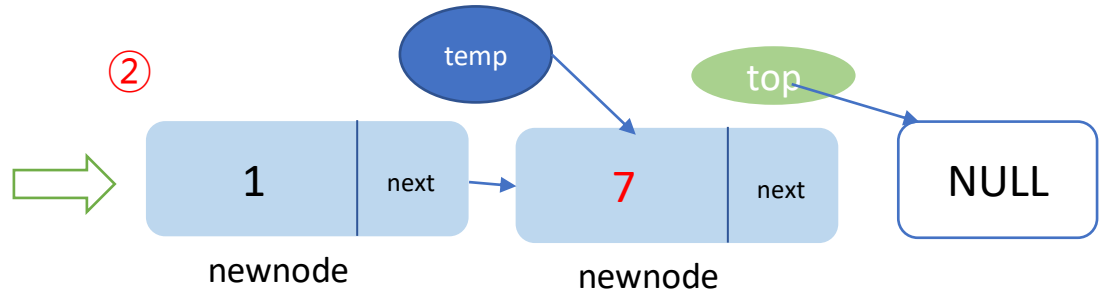
char PopPostfix(PstfixStack *poststcPostfixnode구조체의 자료k)

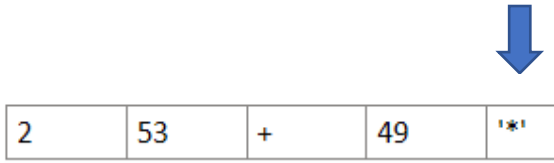
1. 형 temp 포인터 선언
2. 정수자료형인 val선언
3. 만약 poststck의 top이 NULL이라면
   1. ERROR, empty stack...console에 출력
4. 아니라면,
   1. val 은 poststck의 top의 val이다.
   2. tmep는 poststck의 top
   3. poststck의 top 은 poststck의 top의 next이다.
   4. temp의 동적메모리 해제
   5. 반환값 val
5. 반환값 NULL



PopPostfix



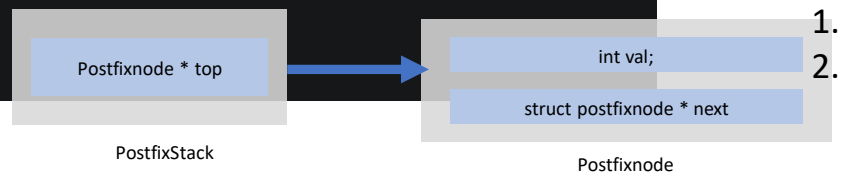




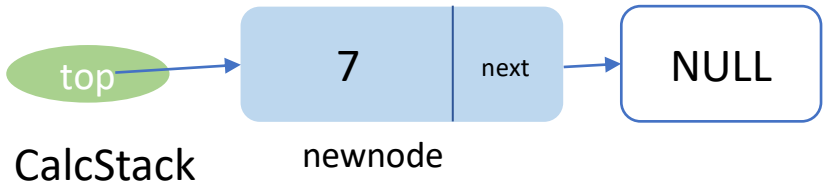
2	53	+	49	'*'	\0

```
i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
        val1=PopPostfix(CalcStack);

        val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

```
PostfixStack * PushPostfix(int val,PostfixStack *poststck)
{
Postfixnode *newnode;
if ((newnode=(Postfixnode*)malloc(sizeof(Postfixnode)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->val=val;
newnode->next=poststck->top;
poststck->top=newnode;
return poststck;
}
}
```



구조체 정리



PostfixStack * PushPostfix(int val,PostfixStack *poststck)

1. Postfixnode구조체의 자료형인 newnode 포인터 선언
2. 만약 newnode는 Postfixnode 타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   NULL반환
3. 아니면라면,
   1. newnode의 val 은 매개변수 val이다
   2. newnode의 next 는 poststck의 top이다.
   3. poststck의 top 는 newnode이다.
   4. 반환값 poststck

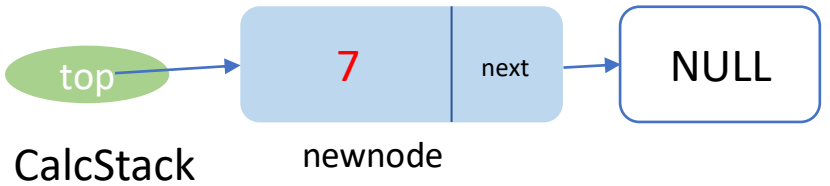
Postfix[]

2	53	+	49	'*'	\0

```
postfix[y]='\0';

//MathStack=FreeAll(MathStack);

/* now calculate the postfix */
/*printf("\nCURRENT POSTFIX=%s\n",postfix);*/

i=0;

CalcStack->top=NULL;
while(postfix[i]!='\x0')
{
    if (isdigit(postfix[i])) {
        /* push to stack */
        CalcStack=PushPostfix(postfix[i]-'0',CalcStack);
    }
    else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*') | (postfix[i]=='/'))
    {
        val1=PopPostfix(CalcStack);

        val2=PopPostfix(CalcStack);


        switch (postfix[i])
        {
            case '+': resultVal=val2+val1;break;
            case '-': resultVal=val2-val1;break;
            case '/': resultVal=val2/val1;break;
            case '*': resultVal=val2*val1;break;
        }

        CalcStack=PushPostfix(resultVal,CalcStack);
    }
    i++;
}

//CalcStack=FreeAll(CalcStack);
LastExpReturn=CalcStack->top->val;
```

```
    }
    WillBreak=0;
    }
}
```

Postfix[5] = \0이기에 while반복문을 빠져나간다.



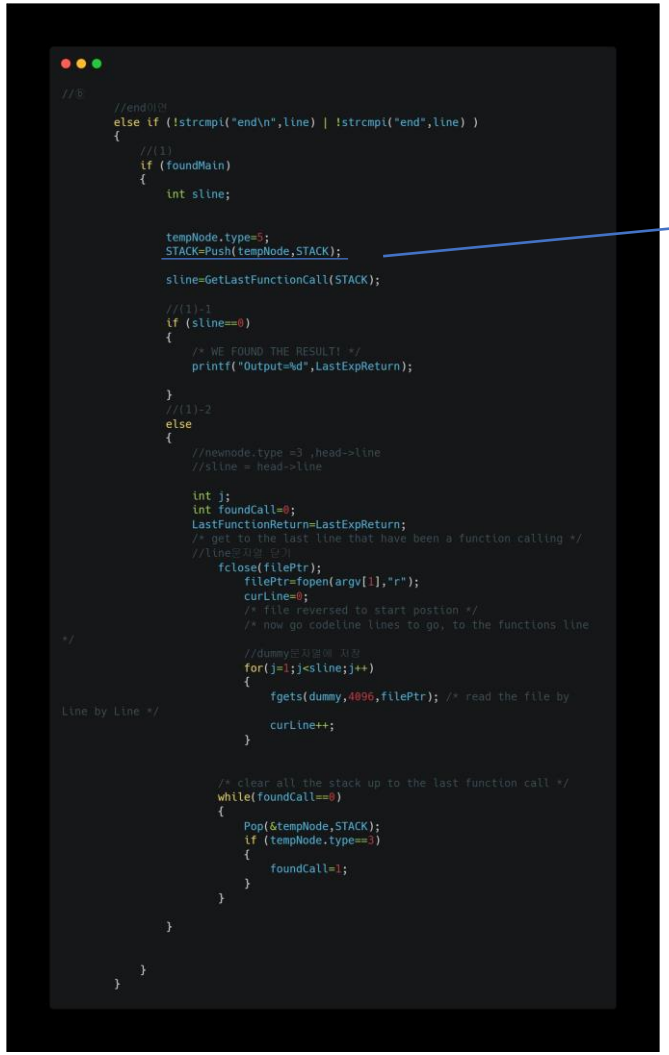
LastExpReturn= CalcStack의 top의 val이기에
LastExpReturn = 7이다.
WillBreak=0;

```
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //fget을 통해 function f(int a)을 읽어 line에 저장 .
    fgets(line,4096,filePtr); /* read the file by Line by Line */
    /* scan for /t characters. get rid of them! */

    // whil문 line[k]이  null 이 아닐 시때까지 반복.
    while(line[k]!='₩0')
    {
        //  4-1) 만약 line[k]가 ₩t(탭키)일 시,
     if (line[k]=='₩t')
     {
        //line[k]는  ' '으로 바꾼다.
       line[k]=' ';
     }
      //k에 1증가
     k++;
    }
    //5. line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을  ' '로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //begin과 line이 동일하다면
    if (!strcmpi("begin₩n",line) | !strcmpi("begin",line)){ ... }


    //ⓑ
    //  //strcmpi :  둘이 동일하면 1(true)  반환하고 동일하지 않으면 0(false)
    //line이 end라면
    else if (!strcmpi("end₩n",line) | !strcmpi("end",line) ){ ... }

    //ⓒ begin과 end가 아니면
    else{ ... }
```

input1.sql(읽을 파일)

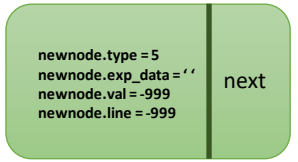
```
function f(int a)
begin
    int b = 6;
    int c = 2;
    ((b+c)/a);
end

function main()
begin
    int a = 1;
    int b = 2;
    int c = 4;
    ((6 + f(c) ) / b);
end
```

line이 end일 때



```
//8
        //end이면
        else if (!strcmpi("end\n",line) | !strcmpi("end",line) )
        {
            //(1)
            if (foundMain)
            {
                int sline;


                tempNode.type=5;
                STACK=Push(tempNode,STACK);

                sline=GetLastFunctionCall(STACK);

                //(1)-1
                if (sline==0)
                {
                    /* WE FOUND THE RESULT! */
                    printf("Output=%d",LastExpReturn);

                }
                //(1)-2
                else
                {
                    //newnode.type =3 ,head->line
                    //sline = head->line

                    int j;
                    int foundCall=0;
                    LastFunctionReturn=LastExpReturn;
                    /* get to the last line that have been a function calling */
                    //line문자열 단계
                        fclose(filePtr);
                            filePtr=fopen(argv[1],"r");
                            curLine=0;
                            /* file reversed to start postion */
                            /* now go codeline lines to go, to the functions line
*/
                            //dummy문자열에 저장
                            for(j=1;j<sline;j++)
                            {
                                fgets(dummy,4096,filePtr); /* read the file by
Line by Line */
                                curLine++;
                            }


                            /* clear all the stack up to the last function call */
                        while(foundCall==0)
                        {
                            Pop(&tempNode,STACK);
                            if (tempNode.type==3)
                            {
                                foundCall=1;
                            }
                        }

                }


            }
        }
```

```
Stack * Push(Node sNode,Stack *stck)
{
Node *newnode;

if ((newnode=(Node*)malloc(sizeof(Node)))==NULL) {
printf("ERROR, Couldn't allocate memory...");
return NULL;
}
else
{
newnode->type=sNode.type;
newnode->val=sNode.val;
newnode->exp_data=sNode.exp_data;
newnode->line=sNode.line;
newnode->next=stck->top;
stck->top=newnode;
return stck;
}
}
```



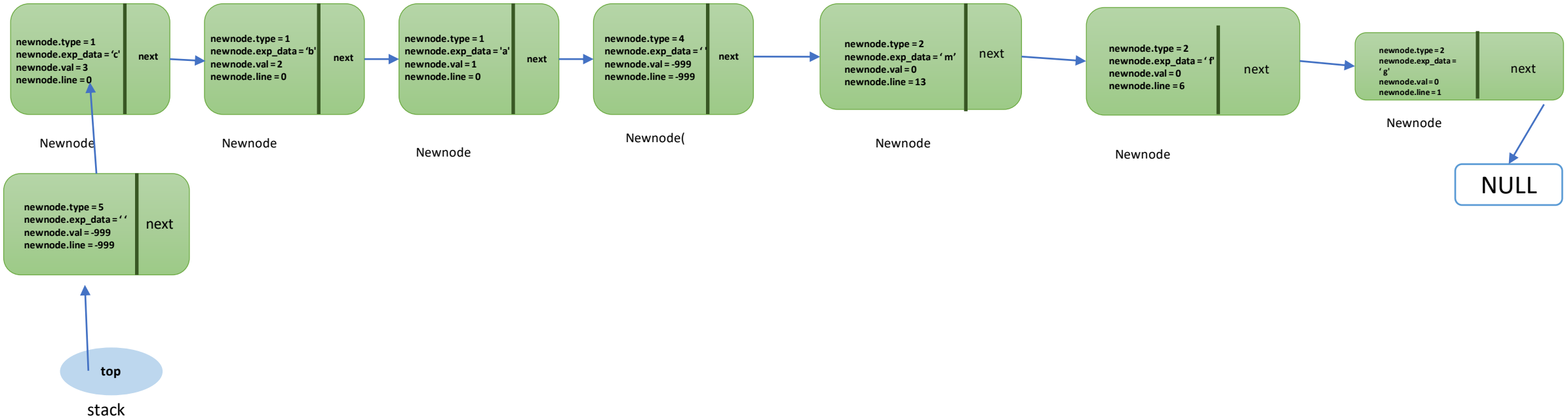
Stack * Push(Node sNode,Stack *stck)

1. Node 자료형의 newnode포인터선언;
2. 만약 newnode는 Node타입크기 만큼메모리를 할당받으며, newnode가 NULL일 경우
   ERROR, Couldn't allocate memory... 출력
   return NULL

3.아니라면

newnode의 type은 5의 타입이 된다.
newnode 의 val은 -999 이된다.
newnode 의 line 은 -999 이 된다.
newnode 의 next 는 stck의 top이 된다.
stck의 top은 newnode이다

반환값 stck


newnode.type = 1
newnode.exp_data = 'c'
newnode.val = 3
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'b'
newnode.val = 2
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'a'
newnode.val = 1
newnode.line = 0
next
Newnode
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
Newnode(
newnode.type = 2
newnode.exp_data = ' m'
newnode.val = 0
newnode.line = 13
next
Newnode
newnode.type = 2
newnode.exp_data = ' f'
newnode.val = 0
newnode.line = 6
next
Newnode
newnode.type = 2
newnode.exp_data = ' g'
newnode.val = 0
newnode.line = 1
next
Newnode
NULL
newnode.type = 5
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
top
stack

line이 end일 때

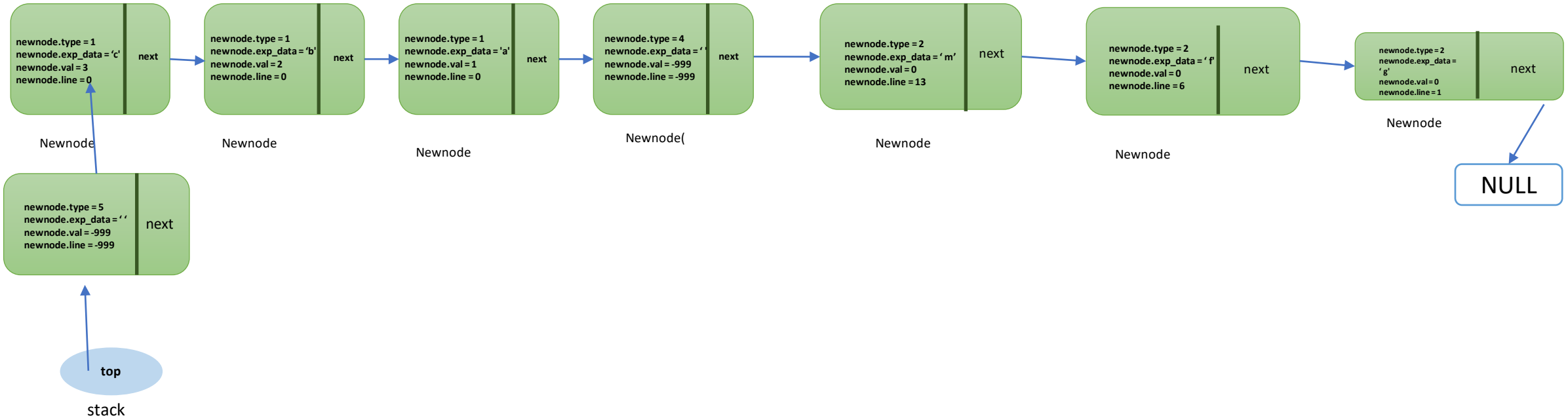
sline=0

GetLastFunctionCall

```
int GetLastFunctionCall(Stack *stck)
{
Node * head;

  if (stck->top == NULL )
  {
    printf("ERROR, empty stack...");
  }
  else
  {
    head=stck->top;
    do
    {
      if ( head->type==3 )
      {
        return head->line;
      }
      else
      {
        head=head->next;
      }
    } while (head->next!=NULL);
  }

  return 0;



}
```

int GetLastFunctionCall(Stack *stck)

1. Node의 구조체의 자료형인 포인터 head
2. 만약 stck의 top 이 NULL이라면,
    1. ERROR, empty stack… 이라고 콘솔 출력
3. 아니라면,
    1. head는 stck의 top이다.
    2. do
        1. head의 type은 3이라면
            1. 반환값 head의 line
        2. 아니라면
            1. head는 head의 next이다.
    3. while (head의 next가 NULL이 아니라면 반복)
4. 반환값 0


newnode.type = 1
newnode.exp_data = 'c'
newnode.val = 3
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'b'
newnode.val = 2
newnode.line = 0
next
Newnode
newnode.type = 1
newnode.exp_data = 'a'
newnode.val = 1
newnode.line = 0
next
Newnode
newnode.type = 4
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
Newnode(
newnode.type = 2
newnode.exp_data = ' m'
newnode.val = 0
newnode.line = 13
next
Newnode
newnode.type = 2
newnode.exp_data = ' f'
newnode.val = 0
newnode.line = 6
next
Newnode
newnode.type = 2
newnode.exp_data = ' g'
newnode.val = 0
newnode.line = 1
next
Newnode
NULL
newnode.type = 5
newnode.exp_data = ' '
newnode.val = -999
newnode.line = -999
next
top
stack

LastExpReturn = 7

```
if (sline==0)
{
    /* WE FOUND THE RESULT! */
    // Output = LastExpReturn 콘솔 출력

    printf("Output=%d",LastExpReturn);
}
```

Output=7
콘솔에 출력

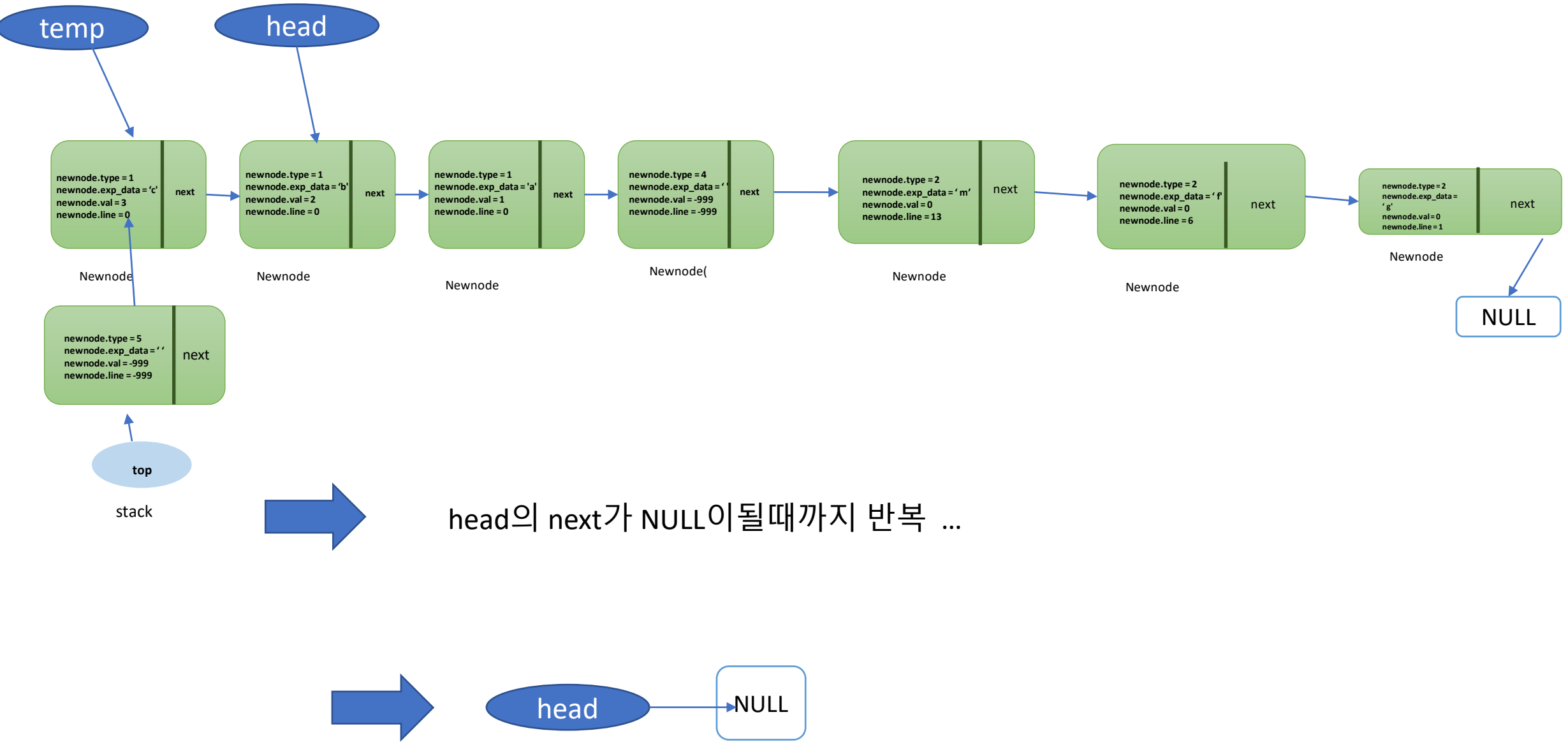
```
}
//filePtr 파일 닫기
fclose(filePtr);
//printAllStack(STACK);

STACK=FreeAll(STACK);

printf("\nPress a key to exit...");
getch();
return 0;
}
```

1.filePtr 파일 닫기
2.STACK= NULL
3. " Press a key to exit..."라고 콘솔에 나온다.
4. 키를 입력받는다.(input)
5. return 0

FreeAll

```
Stack * FreeAll(Stack * stck)
{
Node * temp;
Node * head;

  if (stck->top != NULL )
  {
    head=stck->top;
    do
    {

      temp=head;
      head=head->next;
      free(temp);

    } while (head->next!=NULL);
  }

return NULL;
}
```

Stack * FreeAll(Stack * stck)
1. Node구조체 자료형인 포인터 temp
2. Node구조체 자료형인 포인터 head
3. 만약 stck의 top이 NULL이 아니라면,
    1. head는 stck의top이다.
    2. do
        1. temp는 head이다.
        2. head는 head의 next이다.
        3. temp 동적 메모리 할당 해제한다.
    3. while (head의 next가 NULL이 아니라면 반복)
4. 반환값 NULL



head의 next가 NULL이될때까지 반복 ...

```
//파일의 끝에 도달했는지 여부를 확인
//파일의 끝에 도달하지 못했을 경우에 0 반환
//파일 스트림의 끝을 만날 때까지 반복
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //filePtr의파일을 최대 4095수까지 읽고 line배열에 저장.
    fgets(line,4096,filePtr);

    // whil문 line[k]이 '\0'( null )이 아닐 때까지 반복.
    while(line[k]!='\0')
    {
        //  4-1) 만약 line[k]가 \t(탭키)일 시,
     if (line[k]=='\t')
     {
         //line[k]는 ‘ ‘으로 바꾼다.
        line[k]=' ';
     }
      //k에 1증가
     k++;
    }
    //5. line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을 ‘‘로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  둘이 동일하면 0  반환하고 동일하지 않으면 1
    //! 을 붙음으로써 둘이 동일하면 1(True) 동일하지 않으면 0(False)
    //"begin"과 line이 동일하다면
    if (!strcmpi("begin\n",line) | !strcmpi("begin",line))
    {
        //foundMain == 0이면 False 0이 아니면 True
        //foundMain 이 1이면 if문 실행
        if (foundMain)
        {
            //tempNode.type 는 4이다.
            tempNode.type=4;
            ////STACK은 Push(tempNode,STACK); 에서의 반환값이다.
            STACK=Push(tempNode,STACK);
```

```
        }
    }


    //ⓑ
    // //strcmpi :  둘이 동일하면 0  반환하고 동일하지 않으면 1
    //! 을 붙음으로써 둘이 동일하면 1(True) 동일하지 않으면 0(False)
    //line이 "end"이거나 line이 "end \n"이라면
    else if (!strcmpi("end  \n",line) | !strcmpi("end",line) )
    {
        //(1)

        //foundMain == 0이면 False 0이 아니면 True
        //foundMain 이 1이면 실행
        if (foundMain)
        {
            //sline 정수 선언
            int sline;

            //tempNode의 typ은 5
            tempNode.type=5;
            //STACK은Push(tempNode,STACK)의 반환값
            STACK=Push(tempNode,STACK);

            //sline은GetLastFunctionCall(STACK)의 반환값
            sline=GetLastFunctionCall(STACK);

            //만약 sline이 0이라면
            if (sline==0)
            {
                // Output = LastExpReturn의값을 콘솔 출력
                printf("Output=%d",LastExpReturn);

            }
            //sline==0 이 아니라면
            else
            {
                //정수 j선언
                int j;
                //정수 foundCall는 0
                int foundCall=0;

                //LastFunctionReturn은 LastExpReturn의 값이다
                LastFunctionReturn=LastExpReturn;
                    //filePtr 파일 닫기
                    fclose(filePtr);
                    /// filePtr= argv[1] 파일 읽기모드는 파일열기
                        filePtr=fopen(argv[1],"r");

                        //curLine은 0
                        curLine=0;
```

```
                        //1부터 sline보다 작을때까지 1씩 증가하며 반복
                        for(j=1;j<sline;j++)
                        {
                            //filePtr의파일을 최대 4095수까지 읽을 수 있으며, dummy배
열에 저장.
                            fgets(dummy,4096,filePtr);
                            //curLine 1증가
                            curLine++;
                        }

                    //만일 foundCall이 0일 경우
                    while(foundCall==0)
                    {
                        //Pop(&tempNode,STACK)실행
                        Pop(&tempNode,STACK);
                        //tempNode의 type이 3일 경우
                        if (tempNode.type==3)
                        {
                            //foundCall은 1이다
                            foundCall=1;
                        }
                    }

            }

        }
    }

    //ⓒ begin과 end가 아니면
    else
    {


        // 공백들을 \0으로 바꿈
        //line에서 공백 전의 첫문자
        firstword=strtok(line," ");


        // //strcmpi :  둘이 동일하면 0  반환하고 동일하지 않으면 1
        //! 을 붙음으로써 둘이 동일하면 1(True) 동일하지 않으면 0(False)

        //firstword가 "int"라면
        if (!strcmpi("int",firstword))
        {
            //foundMain ==0
            //foundMain == 0이면 false 0이 아니면 True
            if (foundMain)
            {
                //ttempNode.type 은 1
                tempNode.type=1; /*integer*/
```

```
            //" "공백 기준으로 다음
            //firstword는 공백으로 쪼개진 문자열
            firstword=strtok(NULL," ");

            //tempNode.exp_data의 exp_data = firstword[0]
            tempNode.exp_data=firstword[0];

            //" "공백 기준으로 다음
            //firstword는 공백으로 쪼개진 문자열
            firstword=strtok(NULL," ");

            // //strcmpi :  둘이 동일하면 0  반환하고 동일하지 않으면 1
            //! 을 붙음으로써 둘이 동일하면 1(True) 동일하지 않으면 0(False)
            //firstword가 '='이라면
            if (!strcmpi("=",firstword))
            {
                //" "공백 기준으로 다음
                //firstword는 공백으로 쪼개진 문자열
                firstword=strtok(NULL," ");
            }

            //문자열을 정수 타입 변화 ex) 1
            //tempNode의 val = firstword를 정수 변환
            tempNode.val=atoi(firstword);
            //tempNode의 line은  0
            tempNode.line=0;
            //STACK은 Push(tempNode,STACK)한 값
            STACK=Push(tempNode,STACK);
        }
    }

    // //strcmpi :  둘이 동일하면 0  반환하고 동일하지 않으면 1
    //! 을 붙음으로써 둘이 동일하면 1(True) 동일하지 않으면 0(False)
    // firstword가 "function"이라면
    else if (!strcmpi("function",firstword))
    {
        //tempNode.type는 2이다
        tempNode.type=2;

        //" "공백 기준으로 다음
        //firstword는 공백으로 쪼개진 문자열
        firstword=strtok(NULL," ");

        //tempNode.exp_data는 firstword[0]이다
        tempNode.exp_data=firstword[0];

        //tempNode.line 는 curLine이다
        tempNode.line=curLine;
        //tempNode.val은 0이다.
        tempNode.val=0;
```

```
            //STACK은 Push(tempNode,STACK); 에서의 반환값이다.
            STACK=Push(tempNode,STACK);

            //만약 firstword배열의 0번쨰가 'm'이며 firstword배열의 1번쨰가 'a'이며
firstword배열의 2번쨰가 'i'이며 firstword배열의 3번째까 'n'이라면
            if ( (firstword[0]=='m') & (firstword[1]=='a') & (firstword[2]=='i') &
(firstword[3]=='n') )
            {
                //foundMain 은 1이다.
                foundMain=1;



            }
            //만약 firstword배열의 0번쨰가 'm'이며 firstword배열의 1번쨰가 'a'이며
firstword배열의 2번쨰가 'i'이며 firstword배열의 3번째까 'n'이 아니라면
            else
            {
                //(3)
                //foundMain == 0이면 False 0이 foundMain == 0이 아니면 True
                //foundMain이 0이 아니면 실행
                if (foundMain)
                {

                    //" "공백 기준으로 다음
                    //firstword는 공백으로 쪼개진 문자열
                    firstword=strtok(NULL," ");

                    //tempNode.type 은 1이다
                    tempNode.type=1;

                    //tempNode.exp_data는 firstword[0]이다
                    tempNode.exp_data=firstword[0];

                    //tempNode.val는 CalingFunctionArgVal
                    tempNode.val=CalingFunctionArgVal;

                    //tempNode의line은 0이다;
                    tempNode.line=0;
                    ////STACK은 Push(tempNode,STACK); 에서의 반환값이다.
                    STACK = Push(tempNode, STACK);

                }

            }


        }

        //firstword[0]이 '(' 이라면
        else if (firstword[0]=='(')
        {
```

```
        //foundMain == 0이면 False 0이 foundMain == 0이 아니면 True
        //foundMain이 1이면 실행
        if (foundMain)
            {

            //정수 i는 0
            int i=0;
            //정수 y는 0
            int y=0;

            //MathStack.top은 NULL
            MathStack->top=NULL;




            //lineyedek[i]이 \x0(NULL)'이 아닐때까지 반복
            while(lineyedek[i]!='\x0')
            {
                //lineyedek[i]가 숫자라면 True
                if (isdigit(lineyedek[i])) {
                    //postfix[y에lineyedek[i] 넣기
                    postfix[y]=lineyedek[i];

                    y++;  //y값 1증가

                }

                //lineyedek[i]이 ')'이라면
                else if (lineyedek[i]==')')
                {
                    //0이면 true 1이면 false
                    //0일때
                    if (!isStackEmpty(MathStack) != 0 )
                    {
                        //postfix[y]에 PopOp(MathStack)의 반환값 넣기
                        postfix[y]=PopOp(MathStack);
                        y++; // y값 1증가
                    }
                }

                ////(3)-3-3
                //lineyedek[i]이 '+'이거나 lineyedek[i]이 '-' 이거나 lineyedek[i]이
'+' 이거나 lineyedek[i]이 '/'이라면
                else if ((lineyedek[i]=='+') | (lineyedek[i]=='-') |
(lineyedek[i]=='*') | (lineyedek[i]=='/'))
                {
                    ////(3)-3-3-1
                    /*operators*/
                    //0이면 false 1이면 true
```

```
                    //isStackEmpty(MathStack)의 반환값 0이 아니라면
                    if (isStackEmpty(MathStack) != 0 )
                    {
                        // MathStack은 PushOp(lineyedek[i],MathStack)의 반환값
                        MathStack=PushOp(lineyedek[i],MathStack);
                    }


                    //isStackEmpty(MathStack)의 반환값이 0이라면
                    else
                    {
                        //Priotry(MathStack->top->op)의 반환값이
Priotry(lineyedek[i])의 반환값보다 크거나 같을때
                        if (Priotry(lineyedek[i]) <= Priotry(MathStack->top->op) )
                        {
                            //postfix[y]은 PopOp(MathStack)반환값
                            postfix[y]=PopOp(MathStack);

                            //y값 1증가
                            y++;

                            //MathStack은 PushOp(lineyedek[i],MathStack)이다
                            MathStack=PushOp(lineyedek[i],MathStack);


                        }
                        // Priotry(lineyedek[i])의 반환값이 Priotry(MathStack->top-
>op)의 반환값보다 클 때
                        else
                        {
                            //MathStack은 PushOp(lineyedek[i],MathStack)의 반환값
                            MathStack=PushOp(lineyedek[i],MathStack);
                        }

                    }
                }

                //알파벳 대문자 "A-Z"는 1을 반환.알파벳 소문자 'a-z"는 2를 반환.
                //lineyedek[i]가 영어라면
                else if (isalpha(lineyedek[i])>0)
                {
                    //정수 codeline는 0이다
                    int codeline=0;

                    //정수 dummyint는 0이다
                    int dummyint=0;

                    //정수 retVal는 0이다
                    int retVal=0;
                    //retVal은 GetVal(lineyedek[i],&codeline,STACK);의 반환값
                    retVal=GetVal(lineyedek[i],&codeline,STACK);
```

```
                        //int codeline =1


                        //만일 retVal이 -1이 아니면서 -999이 아니라면
                        if ((retVal!=-1) & (retVal!=-999))
                        {
                            //postfix[y]에retVal+48을한다.
                            postfix[y]=retVal+48;
                            //y에 1증가
                            y++;

                        }
                        ////만일 retVal이 -1이거나 -999이라면
                        else
                        {

                            //만약 LastFunctionReturn은 -999이라면
                            if (LastFunctionReturn==-999)
                            {

                            //정수 j선언
                            int j;

                            //tempNode.type은 3이다.
                            tempNode.type=3;

                            //tempNode.line은 curLine이다
                            tempNode.line=curLine;

                            //STACK은 Push(tempNode,STACK)의 반환값
                            STACK=Push(tempNode,STACK);


                            //CalingFunctionArgVal은
GetVal(lineyedek[i+2],&dummyint,STACK)의 반환값

CalingFunctionArgVal=GetVal(lineyedek[i+2],&dummyint,STACK);


                            //filePtr의 파일을 닫는다.
                            fclose(filePtr);
                            //filePtr은 argv[1]번째를 읽기 모드로 파일을 연다
                            filePtr=fopen(argv[1],"r");
                            //curLine은 0이다.
                            curLine=0;


                            //1부터 codeline보다 작을때까지 1씩 증가하여 반복
                            for(j=1;j<codeline;j++)
                            {
                                //filePtr의파일을 최대 4095수까지 읽을 수 있으며, dummy배
```

```
열에 저장.
                            fgets(dummy,4096,filePtr); /* read the file by Line by
Line */
                            //curLine 1증가
                            curLine++;
                        }

                        //WillBreak은 1이다.
                        WillBreak=1;

                        //whlie문 나가기
                        break;
                        }


                        //만약 LastFunctionReturn은 -999아니라면
                        else

                        {

                        //postfix[y]은LastFunctionReturn+48이다.
                        postfix[y]=LastFunctionReturn+48;
                        y++; // y값 1증가
                        i=i+3; //i는 i+3이다.
                        LastFunctionReturn=-999; //LastFunctionReturn은 -999이다.


                        }
                    }
                    }
                }

                //i에 1 증가
                i++;
            }//while문



            //WillBreak이 0이라면
            if (WillBreak==0)
            {

            //isStackEmpty(MathStack)이 0이라면 반복
            while (isStackEmpty(MathStack)==0)
            {

                //postfix[y]= PopOp(MathStack)의 반환값
                postfix[y]=PopOp(MathStack);
                //y값 1증가
                y++;
            }
```

```
            //postfix[y]에 '\0'을 넣는다.
            postfix[y]='\0';


            // i 는 0
            i=0;

            //CalcStack의 top은 NULL
            CalcStack->top=NULL;

            //postfix[i]이 \x0일때까지 반복
            while(postfix[i]!='\x0')
            {
                //postfix[i]가 숫자라면
                if (isdigit(postfix[i])) {

                    //CalcStack PushPostfix(postfix[i]-'0',CalcStack);의 반환값
                    CalcStack=PushPostfix(postfix[i]-'0',CalcStack);

                }
                //postfix[i]가 '+'이거나 postfix[i]가 '-' 이거나 postfix[i]가 '*'
이거나 postfix[i]가 '/' 이라면
                else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*')
| (postfix[i]=='/'))
                {
                    //val1 은 PopPostfix(CalcStack)의 반환값
                    val1=PopPostfix(CalcStack);

                    //val2 은 PopPostfix(CalcStack)의 반환값
                    val2=PopPostfix(CalcStack);

                    //switch은
                    //postfix[i]은 판단할 값으로 case와 입력값이 있다면 해당 case문
실행
                    switch (postfix[i])
                    {
                        //postfix[i]가 '+'일때 resultVal는 val2+val1의 값이다.
                        //후 break문으로 switch문 빠져나간다.
                        case '+': resultVal=val2+val1;break;
                        //postfix[i]가 '-' 일때 resultVal는 val2-val1의 값이다.
                        //후 break문으로 switch문 빠져나간다
                        case '-': resultVal=val2-val1;break;

                        ///postfix[i]가 '/'일때 resultVal는 val2/val1의 값이다
                        //후 break문으로 switch문 빠져나간다
                        case '/': resultVal=val2/val1;break;

                        //postfix[i]가 '*'일때 resultVal는 val2*val1의 값이다.
                        // //후 break문으로 switch문 빠져나간다
                        case '*': resultVal=val2*val1;break;
                    }
```

```
                    //CalcStack 은 PushPostfix(resultVal,CalcStack)의 반환값
                    CalcStack=PushPostfix(resultVal,CalcStack);
                }
                // i에 1 증가
                i++;
            }


            //LastExpReturn 은 CalcStack의  top의 val이다.
            LastExpReturn=CalcStack->top->val;


            }

            //WillBreak 는 0
            WillBreak=0;
            }
        }
    }


}
//filePtr 파일 닫기
fclose(filePtr);
//printAllStack(STACK);

//STACK은FreeAll(STACK)의 반환값
STACK=FreeAll(STACK);

//Press a key to exit...콘솔에 출력
printf("\nPress a key to exit...");
//키를 입력받는다.(input)
getch();

//반환값 0
return  0;
```

}