중간고사 이후 수업내용 요약

1. 난수(random number) 생성기
   상태도를 이용한 알고리즘
2. 스택을 이용한 알고리즘
   수식의 변경과 계산
3. 노드기반 자료구조 구축 연습
4. 자기참조구조체를 이용한 이진탐색트리 구축 및 활용
5. 정렬알고리즘(버블, 삽입, 퀵, 합병)
6. 자료구조와 알고리즘복습
   알고리즘의 성능분석 big-O 소개

난수 생성

▶ 통계학, 수학, 공학 등의 분야에서 실험데이터로서
특정한 범위의 다양한 난수(random number)가 필요한 경우가 많아 활용도 높음
컴퓨터공학 분야에서도
알고리즘의 적법성 테스트를 다양한 테스트 데이터를 필요로 한다.

▶ 이러한 난수 생성기는 활용도가 높아짐에 따라
편리한 응용 프로그램(앱)으로 개발되어 여러 알고리즘에 활용되고 있다

▶ 활용도를 높이고 편리성을 위하여
사용자 필요에 따라 요구사항을 선택할 수 있는 기능을 제공한다.
- 주어진 범위에서 하나의 숫자 또는 난수 목록을 생성 할 것인지 지정
- 숫자의 범위 설정 가능
- 중복 제외 가능
- 생성한 난수를 정렬하거나 생성하면서 정렬하는 것도 가능하도록 지원함

C함수를 이용한 난수 생성

rand() 함수 : 0~RAND_MAX-1까지의 수 중 랜덤으로 리턴해 준다.
printf("RAND_MAX=%d\n", RAND_MAX);
실행할 때마다 다른 seed값을 가지고 난수를 발생하기 위하여 다음과 같이 srand()를 사용한다
srand((unsigned int)time(NULL));

<stdlib.h>와 <time.h>를 include 해야함

#define random(n) rand() % n

1) random(101)
2) random(101) + 100
3) (char) ('a' + random(26))
4) random(100) + random(100/100.0)

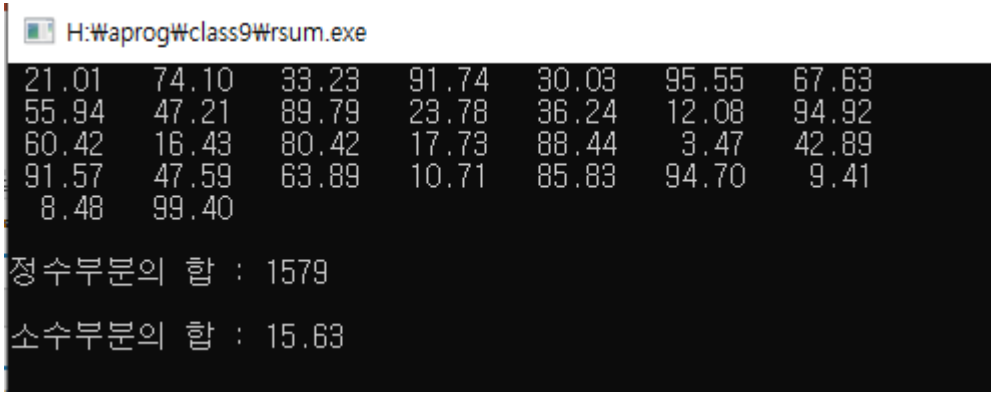
난수생성 함수를 활용한 예제

[예제1] 0.00 - 99.99사이의 실수 난수(random number)를 50개 만들어 배열 rdata에 저장하고 소수 부분의 합과 정수 부분의 합을 구하는 알고리즘을 작성하여 C프로그램으로 코딩하여 확인하시오.

```
#define RNUM 50
#define random(n) rand() % n
main()
{
    float rdata[RNUM], fsum=0.0;
    int i, isum=0, temp;

    srand((unsigned int) time(NULL));
    for (i=0; i < RNUM; i++) {
        rdata[i] = random(100) + random(100)/100.0;
        printf("%6.2f  ", rdata[i]);
        if ((i+1)%7 == 0) printf("\n");
    }
    for (i=0; i < RNUM; i++) {
        temp = rdata[i];
        isum += temp;
        fsum += (rdata[i] - temp);
    }

    printf("\n\n정수부분의 합 : %d\n", isum);
    printf("\n소수부분의 합 : %.3f\n", fsum);
}
```



[예제2] [예제1]의 배열 rdata에 저장된 50개의 데이터를 오름차순으로 정렬하여 출력파일에 저장하는 부분을 첨가하여 C프로그램으로 코딩하여 확인하시오.

```
void selection_sort(float a[], int n);
main(int argc, char *argv[])
{
    float rdata[RNUM], fsum=0.0;
    int i, isum=0, temp;
    FILE *out;

    if (argc != 2) {
            printf("실행인수를 잘못 주었습니다...\n");
            exit(1);
    }
    if ((out = fopen(argv[1], "w")) == NULL) {
      printf("출력파일을 생성할 수 없습니다 \n");
      exit(1);
    }
    srand((unsigned int) time(NULL));
    for (i=0; i < RNUM; i++) {
      rdata[i] = random(100) + random(100)/100.0;
      printf("%6.2f  ", rdata[i]);
      if ((i+1)%7 == 0) printf("\n");
    }
     for (i=0; i < NUM; i++) {
        temp = rdata[i];
        isum += temp;
        fsum += (rdata[i] - temp);
    }
    printf("\n\n: %d\n", isum);
    printf("\n: %.3f\n", fsum);

    selection_sort(rdata, RNUM);
    for (i = 0; i < NUM; i++) {
      fprintf(out, "%6.2f ", rdata[i]);
      if ((i+1)%7 == 0) fprintf(out, "\n");
    }
}
void selection_sort(float a[], int n) {
   int s, m, j;
   float temp;
   for (s = 0; s < n-1; s++) {
     m = s;
     for (j = s+1; j < n; j++)
       if (a[j] < a[m]) m = j;
     temp = a[s];  a[s] = a[m];  a[m] = temp;
   } /* for */
 } /* sort */
```

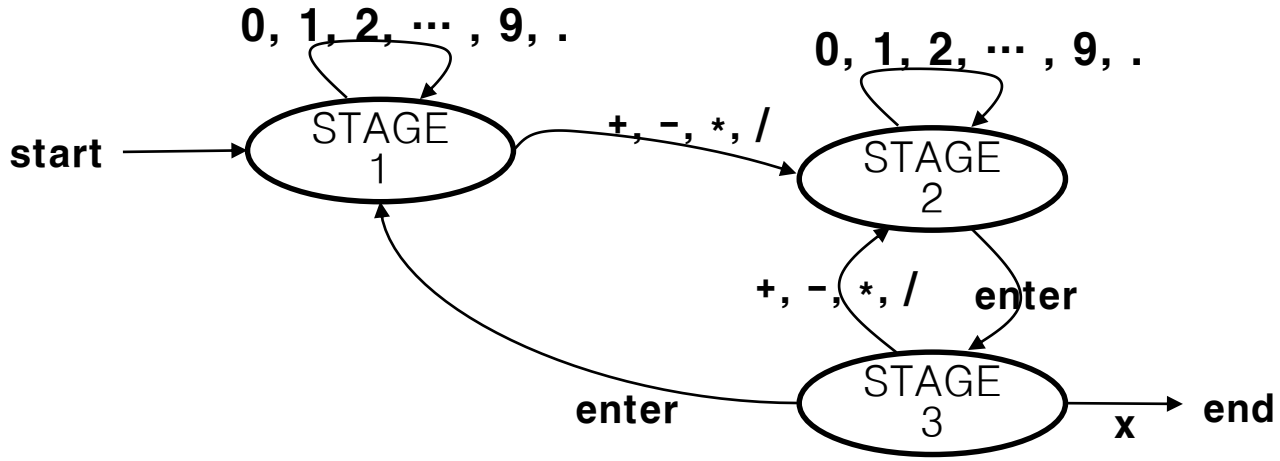
상태도 분석을 통한 알고리즘의 이해(1)

STD(State Transition Diagram)의 활용

- 문제 해결과정안에 여러가지 상태를 포함하는 경우
- 입력에 따라 상태가 변하는 과정을 그림으로 표현하여 알고리즘 작성

[예제1] 정수데이터를 처리하는 단순 계산기 프로그램

0, 1, 2, …, 9와 같은 숫자로 구성된 피연산자와 사칙 연산자를 입력으로 받고 <Enter>를 쳐서 결과를 요구하면 간단한 계산 결과를 준다.



상태도 분석을 통한 알고리즘의 이해(1)

STD(State Transition Diagram)의 활용

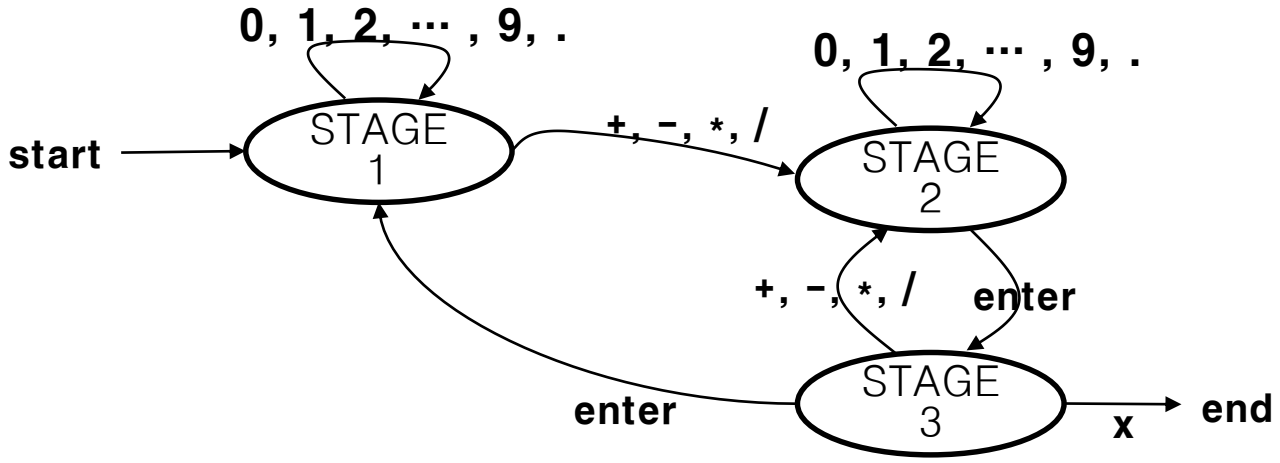
- 문제 해결과정안에 여러가지 상태를 포함하는 경우
- 입력에 따라 상태가 변하는 과정을 그림으로 표현하여 알고리즘 작성

```
#include <stdio.h>
#include <math.h>
#define STAGE1 0
#define STAGE2 1
#define STAGE3 2

void int_calculator(void);
int int_operation(char op, int op1, int op2);
int get_int(int current_operand, char c);
int is_operator(char c);
int is_digit(char c);
main()
{
    printf("계산기 프로그램을 시작합니다.\n");
    printf("정수형의 피연산자와 사칙연산자 중 하나를 입력하시오.\n");
    printf("결과를 받은 후 끝내려면 문자 x를 입력하시오\n");
    int_calculator();
    printf("\n계산기 프로그램을 종료합니다.\n");
}
```

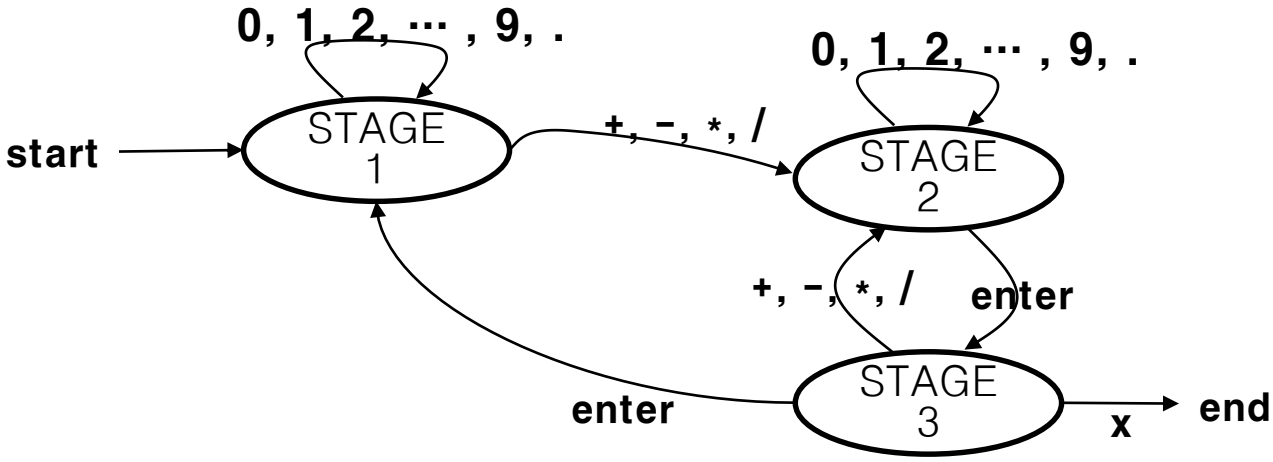
상태도 분석을 통한 알고리즘의 이해(1)

```
void int_calculator(void)
{
    int operand1 = 0, operand2 = 0, result;
    char op, c='0';
    int current_stage = STAGE1;
    while(c != 'x') {
        c = getchar();
        if (current_stage == STAGE1) {
            if (is_digit(c))
                operand1 = get_int(operand1, c);
            else if (is_operator(c)) {
                op = c;
                current_stage = STAGE2;
                operand2 = 0;
            }
        }
```



상태도 분석을 통한 알고리즘의 이해(1)

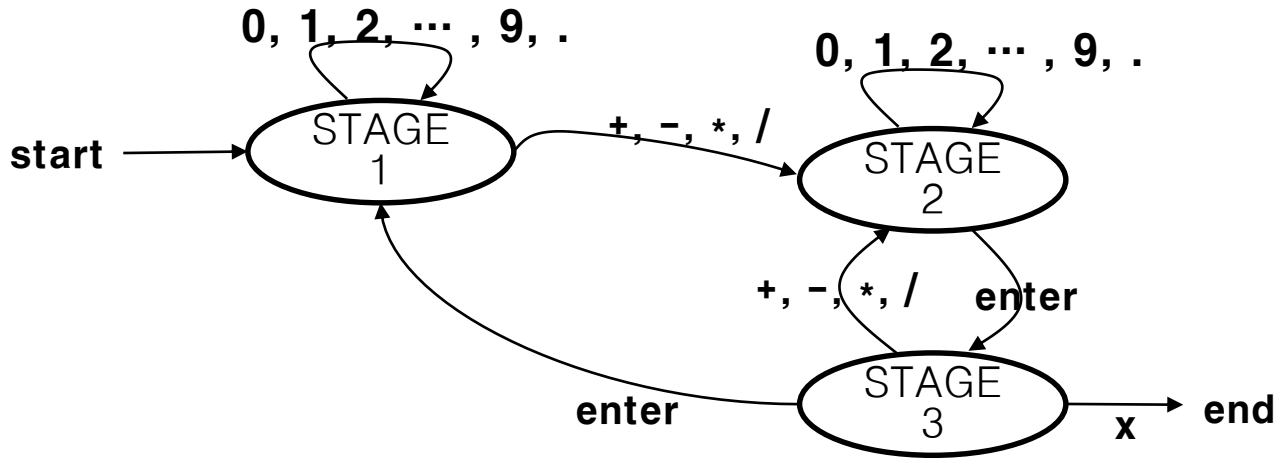
```
else if (current_stage == STAGE2) {
      if (is_digit(c))
         operand2 = get_int(operand2, c);
      else if (c == '\n') {
         result = int_operation(op, operand1, operand2);
         printf("= %d", result);
         current_stage = STAGE3;
      }
 }
```



상태도 분석을 통한 알고리즘의 이해(1)

```
else if (current_stage == STAGE3) {
    if (c == '\n') {
        operand1 = 0;
        current_stage = STAGE1;
    }
    else if (is_operator(c)) {
        op = c;
        current_stage = STAGE2;
        operand1 = result;
        operand2 = 0;
     }
}

//while문과 main() 괄호 닫기
```

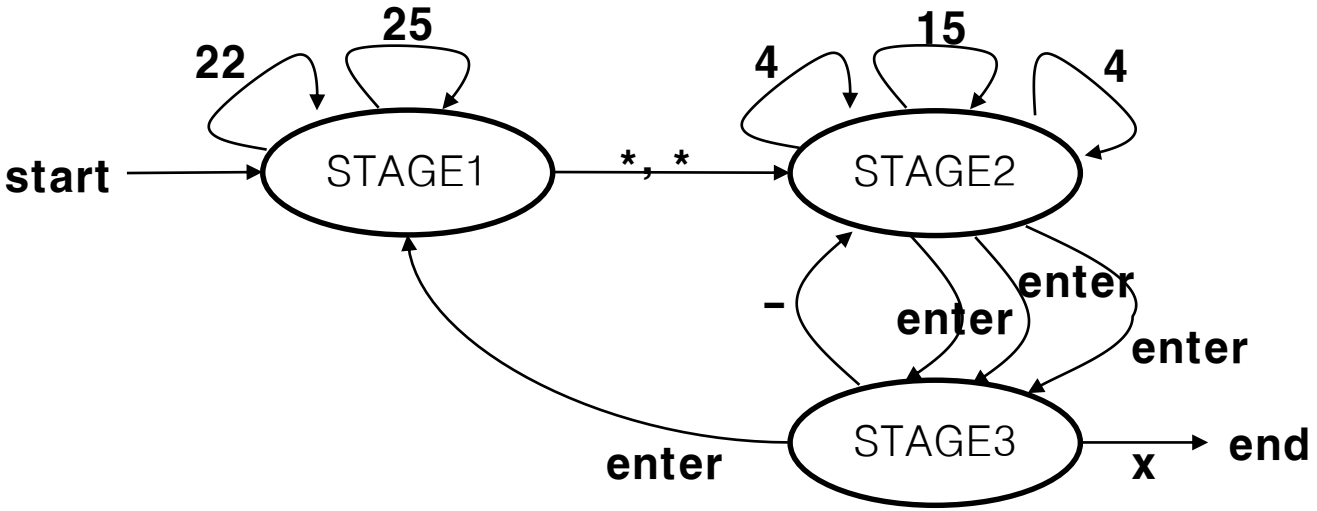


시뮬레이션을 통한 이해

1) 정수 데이터를 처리하는 함수 int_calculator() 시뮬레이션

22*4
=88-15
=73
25*4
=100x

```
int get_int(int current_operand, char c)
{
        int new_value;
        new_value = c - '0';
        current_operand *= 10;
        current_operand += new_value;
        return current_operand;
}
```



기타 함수이해

2) 프로그램에서 사용하는
다음 함수의 기능을 파라미터 값과 결과 값을
예로 들어 코드를 이해해보자.
int int_operation(char op, int op1, int op2);

```
int int_operation(char op, int op1, int op2)
{
    int result;
    switch(op) {
        case '+':
            result = op1 + op2;
            break;
        case '-':
            result = op1 - op2;
            break;
        case '*':
            result = op1 * op2;
            break;
        case '/':
            result = op1 / op2;
                break;
            }
           return result;
}
```

기타 함수이해

2) 프로그램에서 사용하는 다음 함수의 기능을 파라미터 값과 결과 값을 예로 들어 코드를 이해해보자

int get_int(int current_operand, char c);

```
int get_int(int current_operand, char c)
{
        int new_value;
        new_value = c - '0';
        current_operand *= 10;
        current_operand += new_value;
        return current_operand;
}
```

기타 함수이해

2) 프로그램에서 사용하는

다음 함수의 기능을 파라미터 값과 결과 값을

예로 들어 코드를 이해해보자.

int is_operator(char c);

int is_digit(char c);

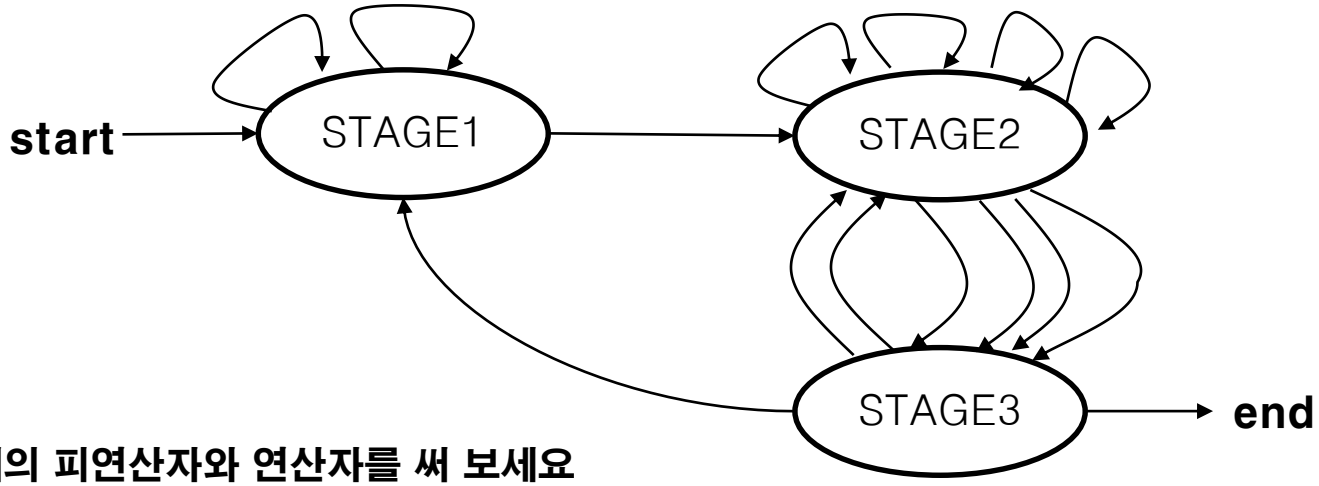
```
int is_operator(char c)
{
    if (c == '+' || c == '-' || c == '*' || c == '/')
        return 1;
    else
        return 0;
}

int is_digit(char c)
{
    if (c >= '0' && c <= '9')
        return 1;
    else
        return 0;
```

상태도 분석을 통한 알고리즘의 이해(2)

[예제2] 실수 데이터를 처리하는 단순 계산기 프로그램
int_calculator() -> real_calculator()
시뮬레이션하면서 소수점 숫자처리를 위해 필요한
status 변수 활용

3756.86/14.5
=259.094
658*0.75
=493.500/12
=41.125+33.86
=74.985x



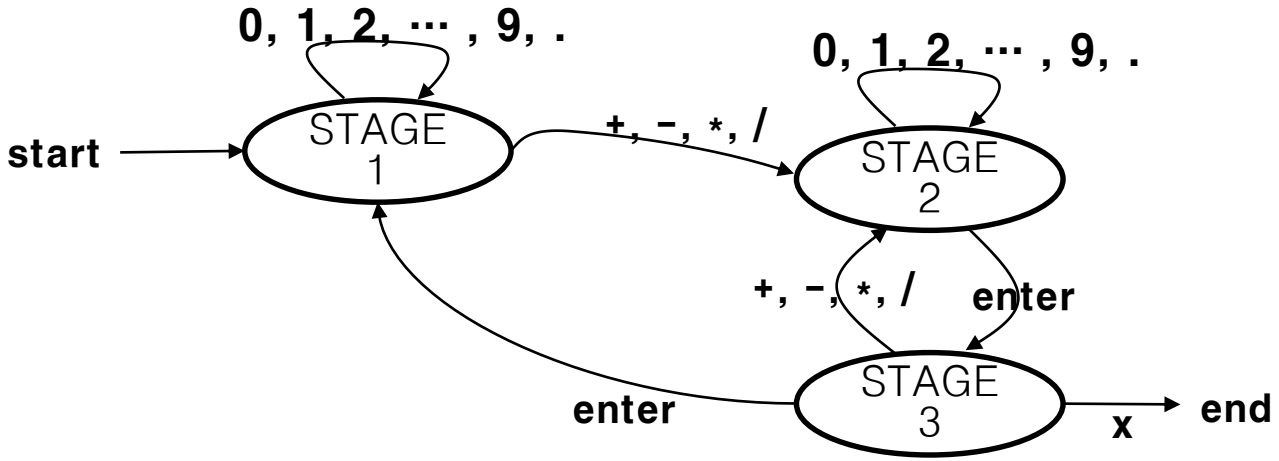
상태도에 위의 예제의 피연산자와 연산자를 써 보세요

상태도 분석을 통한 알고리즘의 이해(2)

```
void real_calculator(void)
{
	double operand1 = 0, operand2 = 0, result;
	int op, c = '0';
	int current_stage = STAGE1, status = 0;
	while (c != 'x') {
		c = getchar();
		if (current_stage == STAGE1) {
			if (is_digit(c)) {
				operand1 = get_real(operand1, c status);
				if (status != 0) status++;
			}
			else if (is_operator(c)) {
				op = c;
				current_stage = STAGE2;
				status = 0;
				operand2 = 0;
			}
			else if (c == '.') status = 1;
		} // STAGE1
```

상태도 분석을 통한 알고리즘의 이해(2)

```
else if (current_stage == STAGE2) {
        if (is_digit(c)) {
                operand2 = get_real(operand2, c, status);
                if (status != 0) status++;
        }
        else if (c == '.') status = 1;
        else if (c == '\n') {
                result = real_operation(op, operand1, operand2);
                printf("= %.3f", result);
                current_stage = STAGE3;
                status = 0;
        }
} // STAGE2
```



상태도 분석을 통한 알고리즘의 이해(2)

```
else if (current_stage == STAGE3) {
        if (c == '\n') {
                operand1 = 0;
                current_stage = STAGE1;
        }
        else if (is_operator(c)) {
                op = c;
                current_stage = STAGE2;
                operand1 = result;
                operand2 = 0;
        }
} // STAGE3
```

상태도 분석을 통한 알고리즘의 이해(2)

함수 int get_int(int current_operand, char c)

-> double get_real(double current_operand, char c, int status)

```
double get_real(double current_operand, char c, int status)
{
	int part1;
	double part2;
	if (status == 0) {
		part1 = c - '0';
		current_operand *= 10;
		current_operand += part1;
	}.
	else {
		part2 = c - '0';
		part2 = part2 * pow(10, -status);
		current_operand += part2;
	}
	return current_operand;
}
```