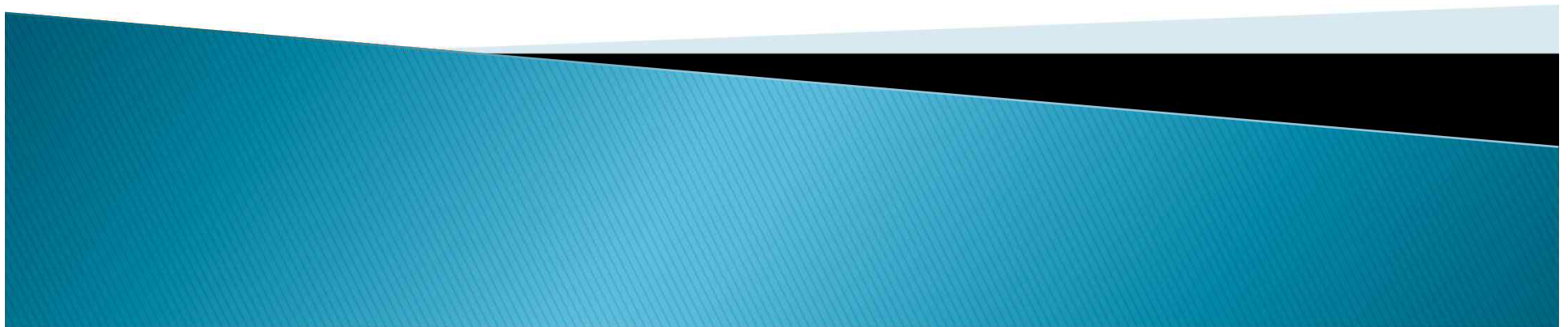


# 프로그래밍랩 - 9주

## 난수, 확률, 통계

정인환교수



# Lab 9-1: 난수

## ▶ 난수 관련 함수

- srand(seed) : 난수 초기화 (seed = time(NULL))
- int rand() : 0 ~ 32767 난수 발생

## ▶ 난수 n개를 발생시키고 합과 평균 출력

```
#include <stdlib.h>
```

```
#include <time.h>
```

```
int i, n, sum=0;
```

```
printf("난수의 개수: ");
```

```
scanf("%d", &n);
```

```
for (i=0; i<n; i++) {
```

```
    r = rand();
```

```
    printf("%d ", r);
```

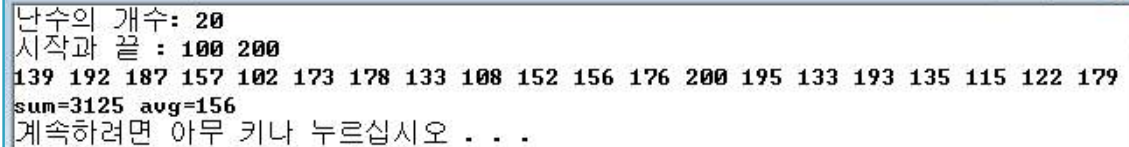
```
    sum += r;
```

```
}
```

```
printf("sum=%d avg=%d\n", sum, sum / n);
```

## ▶ 원하는 범위의 난수 발생 시키기

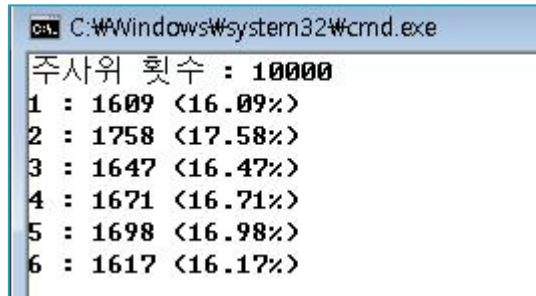
- start <= r <= end
- r = start + rand() % (end - start + 1)
- 0 ~ 100 → r = rand() % 101;



난수의 개수: 20  
시작과 끝 : 100 200  
139 192 187 157 102 173 178 133 108 152 156 176 200 195 133 193 135 115 122 179  
sum=3125 avg=156  
계속하려면 아무 키나 누르십시오 . . .

# Lab 9-2: 확률 계산(주사위)

- ▶ n번 주사위 굴려서 나오는 수 1~6의 빈도수와 확률 출력



```
cmd C:\Windows\system32\cmd.exe
주사위 횟수 : 10000
1 : 1609 <16.09%>
2 : 1758 <17.58%>
3 : 1647 <16.47%>
4 : 1671 <16.71%>
5 : 1698 <16.98%>
6 : 1617 <16.17%>
```

```
int count[6] = {0};
printf("주사위 횟수 : ");
scanf("%d", &ntest);
for (i=0;i<ntest;i++) {
    r = rand()%6; // 0~5 -> 주사위 1~6
                // 번호별 발생 회수 증가(완성할것)
}
// 번호별 결과 출력
```

# Lab 9-2(2): 주사위 확률 조작

- ▶ 주사위 번호들은 1/6(~17%)의 확률임
  - `dice = rand()%6;`
- ▶ 이상한 주사위가 있다고 가정하자
  - 번호가 나오는 확률이 조작되어 있다
  - 1~5 : 각 10%, 6 : 50% 가 나오는 주사위
- ▶ 확률을 조작하는 방법
  - 1, 2, 3, 4, 5, 6 을 구간으로 변경한다
  - 0~9, 10~19, 20~29, 30~39, 40~49, 50~59, 60~69, 70~79, 80~89, 90~99

```
r = rand()%100;  
if (40<= r <=49)  
    dice = 5;  
else if (50 <= r <=99)  
    dice = 6;
```

```
주사위 횟수 : 1000  
정상적인 주사위  
1 : 156 <15.60%>  
2 : 168 <16.80%>  
3 : 176 <17.60%>  
4 : 174 <17.40%>  
5 : 166 <16.60%>  
6 : 160 <16.00%>  
  
이상한 주사위  
1 : 100 <10.00%>  
2 : 105 <10.50%>  
3 : 107 <10.70%>  
4 : 103 <10.30%>  
5 : 107 <10.70%>  
6 : 478 <47.80%>  
계속하려면 아무 키나
```

# Lab 9-3: 모의 성적 데이터 발생1

- ▶ 성적 처리 프로그램을 위한 테스트 데이터를 생성하고자 함
- ▶ n명의 성적을 발생 시키고 각 성적별 빈도수와 %를 출력한다
  - $0 \leq \text{score} \leq 100$
  - $\text{score} = \text{rand}() \% 101$
  - A+(>=95), A(>=90), B+(>=85), .. D(>=60), F(<60)
  - 단순히 0~100을 동일한 확률로 발생한 경우
  - 테스트 데이터로 적합하지 않음 (n=10000 F == 60.05%)

```
1 39 54 70 11 60 62 8 37 59 74 72 76 81 1 56 92 61 42 76 62 85 86 38
n=10000 평균 = 49.90
A+ : 622 ( 6.22%)
A  : 472 ( 4.72%)
B+ : 490 ( 4.90%)
B  : 473 ( 4.73%)
C+ : 524 ( 5.24%)
C  : 447 ( 4.47%)
D+ : 475 ( 4.75%)
D  : 492 ( 4.92%)
F  : 6005 (60.05%)
계속하려면 아무 키나 누르십시오 . . .
```

# Lab 9-4: 모의 성적 데이터 발생2

- ▶ 성적분포가 주어졌을 때 모의 데이터 발생시키기
  - A+:10%, A:10%, B+:20%, B:25%, C+:5%, C:10%, D+:5%, D:3%, F:2%
- ▶ 주사위 확률 조작 방법과 같은 방법으로 A+ ~ F 에 해당하는 점수 발생
- ▶ 각 학점별 % 분포가 학생수(정수)로 환산할 때 정확하게 일치하지 않을 수 있음
  - 각 학점별 배정 가능한 최대 인원수를 미리 계산해 놓고 성적을 발생시킬때 인원 check를 한다.
  - 소수점으로 나오는 경우 정수에서 모자라는 부분 조정
  - n = 150 인 경우
    - % 로만 계산하면 148명 (소수점 문제)
      - A+ 15, A 15, B+ 30, B 37.5, C+ 22.5, C 15, D+ 7, D 4, F 3
    - B 37.5명 → 38명, C+ 22.5 → 23명 으로 조정
      - A+ 15, A 15, B+ 30, B 38, C+ 23, C 15, D+ 7, D 4, F 3

# Lab 9-4: 모의 성적 데이터 발생2

## ▶ 성적 발생 Algorithm

```
int grade; // 0~8, A+ ~ F
int count[9]; // 0 ~ 8 A+ ~ F 인원수
int maxcount[9]; // 미리 계산된 최대 인원
int start[9]={95, 90, 85, 80, 75, 70, 65, 60, 0}; //점수시작
int end[9]={100, 94, 89, 84, 79, 74, 69, 64, 59}; //점수끝
for (i=0;i<9;i++)
    // maxcount[i] 를 계산해 둔다
for (i=0;i<nstudents;i++) {
    do {
        prob = rand()%100; // 확률 변수로 사용
        if (0<= prob <=9) grade = 0; // A+
        else if (0<=10 prob <=19) grade = 1; // A
        ... // 이미 해당 학점의 점수를 모두 발생하였다면
            // 다른 학점을 발생할 때 까지 반복한다.
    } while (count[grade]+1 > maxcount[grade]); //인원 check
    // A+의 경우 start=95 end=100
    score = start[grade] + rand()%(end[grade]-start[grade]+1);
    count[grade]++; // 학점별 count를 증가시킨다
```

# Lab 9-4: 모의 성적 데이터 발생

- ▶ 분포와 인원을 고려한 모의 데이터 발생 예

```
150
85 78 100 91 71 85 70 70 81 74 91 82 83 93 87 78 60 88 71 89 88 87 90 77 71 81 7
92 84 82 91 83 87 61 83 66 86 80 85 72 96 77 81 86 79 84 90 4 88 77 75 100 83 8
82 78 86 95 86 67 20 72 98 95 81 84 65 43 84 84 75 79 77 87 95 88 84 89 82 82 8
94 88 83 85 96 80 93 90 80 76 88 76 68 92 79 81 81 81 83 75 80 84 80 92 91 68 6

n=150 평균 = 81.26
A+ : 15 < 10.00%
A : 15 < 10.00%
B+ : 30 < 20.00%
B : 38 < 25.33%
C+ : 23 < 15.33%
C : 15 < 10.00%
D+ : 7 < 4.67%
D : 4 < 2.67%
F : 3 < 2.00%
계속하려면 아무 키나 누르십시오 . . .
```



# Lab 9-5: 문장의 알파벳 분포 조사

- ▶ input.txt 를 읽어서 알파벳들의 문자수를 count하여 분포를 표시 (속성 < input.txt)
- ▶ 입력은 끝(EOF) 검사하는 방법
  - EOF (End of File) : 파일의 끝 또는 key CTRL-Z

```
int count[26];  
while ((c=getchar())!=EOF) { // 입력의 끝이 아니면..  
    total++;  
    if (isalpha(c)) { // 알파벳만 검사  
        alpha++;  
        count[toupper(c) - 'A']++;  
    }  
}
```

```
전체문자수=1084 알파벳수=849  
A:73 < 6.73%> B:18 < 1.66%> C:24 < 2.21%> D:18 < 1.66%> E:100 < 9.23%>  
F:21 < 1.94%> G:12 < 1.11%> H:50 < 4.61%> I:72 < 6.64%> J:0 < 0.00%>  
K:2 < 0.18%> L:37 < 3.41%> M:32 < 2.95%> N:70 < 6.46%> O:59 < 5.44%>  
P:23 < 2.12%> Q:1 < 0.09%> R:43 < 3.97%> S:50 < 4.61%> T:77 < 7.10%>  
U:26 < 2.40%> V:8 < 0.74%> W:16 < 1.48%> X:3 < 0.28%> Y:14 < 1.29%>  
Z:0 < 0.00%>  
계속하려면 아무 키나 누르십시오 . . .
```

# Lab9-ACM: 괄호 검증 - VPS

- ▶ 중첩된 괄호 ( ) 의 유효성 검사
- ▶ 올바른 괄호 문자열 VPS(Valid Parenthesis String)
  - `((()))((()))` → YES VPS
  - `((())((()))())` → NO VPS
- ▶ 괄호로만 되어있는 입력문자열이 VPS 이면 YES  
아니면 NO
- ▶ `(수 == )수` → YES
- ▶ 문자열 입력
  - `char buf[51];`
  - `scanf()` 또는 `gets()`

Input	Output
8	NO
<code>((()))()</code>	NO
<code>((())())()</code>	YES
<code>((()))((()))</code>	NO
<code>((())((()))((()))())</code>	YES
<code>((())((()))((()))())</code>	NO
<code>((())((()))((()))())</code>	NO
<code>((())((()))()</code>	NO
<code>((())</code>	
<code>)()</code>	

The 37<sup>th</sup> Annual ACM  
International Collegiate Programming Contest  
Asia Regional – Daejeon  
Nationwide Internet Competition



## Problem G

### 괄호(Parenthesis)

괄호 문자열(Parenthesis String, PS)은 두 개의 괄호 기호인 '(' 와 ')' 만으로 구성되어 있는 문자열이다. 그 중에서 괄호의 모양이 바르게 구성된 문자열을 올바른 괄호 문자열(Valid PS, VPS)이라고 부른다. 한 쌍의 괄호 기호로 된 "( )" 문자열은 기본 VPS 이라고 부른다. 만일  $x$  가 VPS 라면 이것을 하나의 괄호에 넣은 새로운 문자열 " $(x)$ "도 VPS 가 된다. 그리고 두 VPS  $x$  와  $y$  를 접합(concatenation)시킨 새로운 문자열  $xy$ 도 VPS 가 된다. 예를 들어 "( ( ) ) ( )"와 "( ( ( ) ) )" 는 VPS 이지만 "( ( ) (", "( ( ) ) ( ) )", 그리고 "( ( )" 는 모두 VPS 가 아닌 문자열이다.

여러분은 입력으로 주어진 괄호 문자열이 VPS 인지 아닌지를 판단해서 그 결과를 YES 와 NO 로 나타내야 한다.

### 입력(Input)

입력 데이터는 표준 입력을 사용한다. 입력은  $T$ 개의 테스트 데이터로 주어진다. 입력의 첫 번째 줄에는 입력 데이터의 수를 나타내는 정수  $T$ 가 주어진다. 각 테스트 데이터의 첫 번째 줄에는 괄호 문자열이 한 줄에 주어진다. 하나의 괄호 문자열의 길이는 2 이상 50 이하이다.

### 출력(Output)

출력은 표준 출력을 사용한다. 만일 입력 괄호 문자열이 올바른 괄호 문자열(VPS)이면 "YES", 아니면 "NO"를 한 줄에 하나씩 차례대로 출력해야 한다.

다음은 6 개의 데이터를 가지는 입력과 출력의 예를 보여주고 있다.

Sample Input	Output for the Sample Input
6	NO
( ) ( )	NO
(( ( ) ( ) ) ( )	YES
( ) ( ) ) ( ( ) ) )	NO
(( ( ) ( ) ( ) ) ) ( ( ( ( ) ) ) ) ( )	YES
( ) ( ) ( ) ( ) ( ) ( ) ( ) ( )	NO
( ) ( ( ( ) ) ) ( ) (	