



# 고급 SW 실습 I

## Random Sequence Generation on a Specific Distribution (실습 자료)

CSE4152

서강대학교 컴퓨터공학과



# 실습 안내

## ◆ 실습 결과물 확인

- ◆ 프로그램 완성 후 담당 조교에게 확인을 받아야 하고 동시에 이를 사이버 캠퍼스 해당 제출함에 제출하여야 한다.
- ◆ 제출할 파일 이름은 **snnnnnnnL05.cpp**로 하여야 한다.
  - ◆ 여기서, **nnnnnnn**은 자신의 학번 뒤 6자리.
- ◆ 실습 결과 검사
  - ◆ 담당 조교가 결과를 검사하면서 제대로 알고 작성했는지 몇가지 작성 내용에 관한 질문을 할 수 있다.
  - ◆ 평가 사항이므로 이에 답을 제대로 못하면 감점할 수 있다.
  - ◆ 그러니, 프로그램을 작성할 때 내용을 이해하며 작성하여야 한다(질문이 있으면 주저 말고 조교에게 문의할 것)

**(주의)** 만일 파일의 nnnnnnn을 자신의 학번 뒤 6자리로 바꾸지 않고, 그냥 snnnnnnnL05.cpp 등으로 제출하면 **0점 처리**한다.



## ◆숙제가 있을 경우

- ◆제출 파일 이름, 마감일 등을 지정해 줄 것이다.

## ◆제출 마감

- ◆실습, 숙제 모두 제출 마감일이 지정되어 있다.
- ◆Late 제출은 허용하지 않는다. 사이버 캠퍼스가 효과적으로 late 제출을 받지 않을 것이다.

## ◆실습 시 검사를 못 받은 경우

- ◆일단 완성하여 실습 프로그램 제출함에 마감 전 제출한다.
- ◆다음 실습 시간에 담당 조교의 양해하에 잠깐 시간을 내어 이전 주 실습 결과를 검사 받을 수 있다(감점이 있을 것임).

## ◆실습 프로그램을 제출했는데 검사를 받지 않은 경우

- ◆반드시 검사를 받아야 한다.
- ◆그렇지 않으면 제출물을 무효화 할 수 있다.



# Visual Studio Project 생성

## ◆ 생성 내용 및 방법

◆ VS 콘솔 프로그램을 위한 프로젝트(기존 생성한 프로젝트를 사용해도 무방하다).

### ◆ VS2017 실행<sup>(1,2)</sup>

◆ 파일 → 새로 만들기 → 프로젝트 → Visual C++ → 기타 → 빈 프로젝트 선택

◆ 프로젝트 이름 → 폴더 선택 → 확인

### ◆ Source File 폴더

◆ 프로젝트 폴더가 있는 위치에 source file들을 저장할 폴더를 하나 만든다.

◆ 이 폴더에 자신이 작성한 프로그램과 입력 데이터를 저장하면 편리하다.

(1) VS2015, VS2019도 사용 가능할 것이다.

(2) X64에서의 작업은 같은 프로젝트에서 x64로 바꾸어 수행할 수 있다.



# 실습 프로그램 구성 및 입출력

## ◆ 실습 프로그램

◆ 두 개의 파일이 주어진다.

### ◆ function.h

- Include files, 필요한 정의 등을 포함한다.
- 학생이 작성하지 않아도 되는 함수를 포함한다.

### ◆ snnnnnnL05\_std.cpp

- 실습 및 숙제인 네 가지 방법에 대한 함수, 그리고 main 외 필요한 함수들이 정의되어 있다.

◆ 사전 게시한 파일들을 다운받아 이름을 바꾸어 사용한다<sup>(1)</sup>

◆ functionL05.h → 그대로 사용

◆ snnnnnnL05\_std.cpp → snnnnnnL05.cpp (실습 완료 후 이 파일만을 사이버 캠퍼스에 제출한다)

숙제의 경우 이 파일을 snnnnnnH05.cpp로 바꿔 제출한다.

(1) nnnnnn은 자신의 학번 뒤 6자리

## ◆ 입력 파일

◆ 두 가지 입력 파일이 필요하다.

◆ stdin으로 redirection을 통하여 읽어야 할 파일(1)

test case 한 셋	1	← Test 개수
	30000 100 1.0	← 생성할 난수 개수 (30000), histogram 크기 (100) 및 범위 (0.0~1.0)
	1 50 1.0e-5 1.0e-5 1	← 생성 방법(1), 최대반복회수(50), $\epsilon$ , $\delta$ , 중간결과표시:1, 아니면 0
	01_IN_sample.txt	← sample seq 파일(입력)
	01_pdf.txt	← pdf 파일(출력)
	01_cdf.txt	← cdf 파일(출력)
	01_output_bi.txt	← 생성된 난수 파일(출력)
	01_histogram_bi.txt	← Histogram 파일(출력)

(1) 폴더 '입력파일'에 lab05\_01\_IN.txt 등 네 개의 파일을 제공한다.



## ◆ 파일에서 직접 읽어야 할 파일

◆ Sampled sequence 파일로 그 형식은 다음과 같다.

```
100 5.797980
```

```
7.000000 435.000000
```

```
12.797979 429.264679
```

```
18.595959 421.329529
```

```
24.393940 411.930939
```

```
. . .
```

← 샘플 개수(100), 샘플간 x축 간격

100 개의 샘플 ( $x_i, y_i$ )



## ◆ 출력 파일

- ◆ stdin으로 읽은 parameter 파일에 열거한 파일들이 출력된다.
- ◆ pdf와 cdf 파일은 입력인 sample sequence 파일과 동일한 format으로 출력된다.
- ◆ 생성된 난수 파일 형태는 첫 줄에 총 난수 개수, 다음 줄부터 한 줄에 한 개씩 난수가 기록되어 있다.
- ◆ Histogram 파일은 다음과 같은 형식으로 출력된다

```
100 0.010000 30000 3191.00 ms
0.005000 500
0.015000 487
0.025000 469
0.035000 456
. . .
```

100 개의 bin  
(bin 중간 값, 개수)

← 빈 개수(100), 빈 간 x축  
간격, 난수 개수, 계산시간



## ◆ 프로그램 실행 (실행 파일을 첨부하였으니 실행해보자)

### ◆ 다음과 같이 명령 프롬프트에서 실행한다

```
snnnnnnnL05 < in.txt > out.txt <ent>
```

**in.txt** : 입력 parameter 파일 (lab05\_01\_IN.txt 등)

**out.txt** : 프로그램 실행 중간 진행 사항 (showSteps=1인 경우)

### ◆ out.txt 예(showSteps=1인 경우)

난수 순서

n(	0)	xn1	f(xn1)	U = 0.3319	← U 값
0	2.50000000000000e-01		3.5411556498160524e-03		
1	2.536973767417e-01		6.0169870885751209e-06		
n(	1)	xn1	f(xn1)	U = 0.6126	
0	7.50000000000000e-01		2.6100465826734975e-03		
1	7.475155324977e-01		2.5884719100033315e-05		
2	7.475404145851e-01		5.1428704883615950e-07		
...					

$X$  값                       $|F_X(X) - U|$  값

- 위에서 보인 것처럼 showSteps=1이면 풀이 방법의 수렴 정도를 파악할 수 있다.
- 풀이 방법을 작성할 때 showSteps=1이면 위와 유사한 출력이 되도록 하여야 풀이가 제대로 되고 있는지 확인할 수 있다.



## 실습 프로그램 작성

### ◆ 개요 및 주의사항

- ◆ 프로그램 `snnnnnnnL05_std.cpp`에 작성해야 할 부분을 표시해 두었다. 이를 작성하여 프로그램을 완성하자.

### ◆ `main( )` 함수

- ◆ Histogram을 생성하는 부분을 작성한다.

### ◆ `pdfGeneration( )`

- ◆ 입력된 sample sequence를 pdf form이 되도록 normalize한다.
- ◆ 먼저  $x$  좌표의 범위를 0.0 ~ 1.0로 변환한다.
- ◆ 다음, 사다리꼴 공식으로 전체 적분 값을 구한 후 각 샘플 값을 이 값으로 나눈다.

### ◆ `cdfGeneration( )`

- ◆ 사다리꼴 공식을 사용하여 cdf  $F_X(x)$  값을  $x_1$ 부터 순차적으로 계산한다( $F_X(0) = 0.0$ ,  $F_X(x_{n-1}) = 1.0$ 이 되어야 한다)



## ◆ interpolationF( )

- ◆ Linear interpolation을 통하여  $F_X(X) - U$  값을 계산한다.
- ◆ 배열 pdfX[ ]와 cdfY[ ]를 사용한다.

## ◆ interpolationFD( )

- ◆ Linear interpolation을 통하여  $p_X(X)$  값을 계산한다<sup>(1)</sup>.
- ◆ 배열 pdfX[ ]와 pdfY[ ]를 사용한다.

## ◆ genRandN\_Bisection( )

- ◆ 이 방법은 수렴 속도가 느리지만 반드시 해를 찾는다.
- ◆ 근을 찾는 구간도 [0.0 ~ 1.0]으로 하면 된다.
- ◆ 어떤  $X$  값에 대해  $F_X(X) - U$ 를 구하는 것은  $F_X(X)$ 를 linear interpolation에 의하여 계산하고 이 값에서  $U$ 를 빼서 구한다.
- ◆ 나머지는 4주차 실습에서의 방법과 동일하다.

(1)  $F'_X(x) = p_X(x)$ 로 Newton-Raphson 방법에 필요하다.



## ◆ genRandN\_NewtonRaphson( )

- ◆  $F_X(X) - U = 0$ 인  $X$ 를 Newton-Raphson 방식으로 계산한다.
- ◆  $F_X(x)$  값과  $F'_X(x)$  값은 앞에서 언급한 함수로 계산한다.
- ◆ 이 방법을 위해서는 초기 값 설정이 필요한데, 이를 어찌 구할지 생각해서 적용해보자.
  - ◆  $F_X(X)$ 가 단조 증가 함수이고  $X$ 의 범위가  $[0,1]$  사이이므로 heuristic하게 초기값을 설정할 수 있을 수 있다.
  - ◆ 혹은, bisection 방법을 몇 번 적용하여 근의 범위를 줄인 후 초기값을 선택할 수도 있겠다.
- ◆ 어떤 pdf에 대해서는 초기 값을 아무리 잘 선택해도 어떤 구간에서 이 방법은 발산할 수 있다.
- ◆ 따라서, 입력 파라미터 파일을 Newton-Raphson 방법으로 구하는 테스트 케이스 하나만으로 설정하고, 생성되는 중간 출력 상황을 살펴 그 원인을 파악하자.



## ◆ 엑셀을 통한 결과 보기

- ◆ Microsoft EXCEL을 사용하여 pdf, cdf, histogram 등을 plot해보자.

- ◆ Plotting 하는 방법은 담당 조교가 설명해 줄 것이다.

## ◆ 다른 샘플에 대한 테스트

- ◆ 게시 자료 중 폴더 SampleGen에 sampled sequence를 생성할 수 있는 프로그램이 있다.

- ◆ 담당 조교가 이 사용법을 알려줄 것이다(아주 쉽다).

- ◆ 이를 통하여 다른 sampled sequence를 만들어 Newton-Raphson 방법의 발산 여부를 체크하자(발산하게 만들기 위해서는 생각이 필요하다).



- ◆ `genRandN_Secant( )`
  - ◆ 속제로 남긴다.
- ◆ `genRandN_Improved`
  - ◆ 속제로 남긴다



## 실습 결과 데모 및 제출

### ◆ 실습 결과 데모

- ◆ 실습 프로그램 완료 후 담당 조교에게 검사를 받는다.

### ◆ 실습 결과 제출

- ◆ snnnnnnL05.cpp 파일을 사이버 캠퍼스의 해당 제출함에 제출한다.

### ◆ 실습을 완료하지 못한 경우

- ◆ 실습 결과 제출 마감 전까지 완료하여 사이버 캠퍼스의 해당 제출함에 제출한다.
- ◆ 6주차 실습 시간에 검사를 받는다.