



PROCESO PERSONAL DE DESARROLLO DE  
SOFTWARE 2

VERANO 2016

Apellido Paterno	Gutiérrez
Apellido Materno	Canto
Nombre(s)	Juan Alberto
Matrícula	24400063
Descripción de tarea	Programa )A
Fecha	19 de Julio de 2016

### Checklist de Correcciones.

Status	Número	Descripción
	1	
	2	
	3	
	4	
	5	
	6	
	7	
	8	
	9	



PROCESO PERSONAL DE DESARROLLO DE  
SOFTWARE 2

VERANO 2016

**Forma PSP2 Project Plan Summary.**

### PSP2.1 Project Plan Summary - Program 3A

Student 94419-JUAN ALBERTO GUTIER Date 11/11/16  
 Program Chi-square Test Program# 3A  
 Instructor JORGE RAFAEL AGUILAR CISI Language C

	Plan	Actual	To Date	To Date%
<b>Summary</b>				
LOC/Hour	29.61	26.50	24.78	
Planned Time	274		1367	
Actual Time		725	2302	
CPI (Planned/Actual Time)			0.59	
% Reused	23.8	0.0	15.5	
% New Reused	0.0	0.0	0.0	
Test Defects/KLOC	6.99	3.13	4.95	
Total Defects/KLOC	87.41	25.00	54.46	
Yield %	92.0	75.0	87.9	
<b>1 Appraisal CCR</b>	14.1	10.3	12.3	
<b>2 Failure CCR</b>	5.6	6.6	6.1	
<b>CCR A/F Ratio</b>	2.54	1.55	2.01	

#### Program Size (LOC)

Base(B)	105	95	
Deleted(D)	30	20	
Modified(M)	50	15	
Added(A)	85	305	
Reused(R)	50	0	118
Total N&C (N)	135	320	606
Total LOC(T)	210	340	760
Total New Reused	0	0	0
Total Object LOC(E)	128.6	249	533
UPI (70%)	186		
LPI (70%)	85		

#### Time in Phase (min.)

Planning	42	172	287	19.5
Design	121	339	667	45.4
Design Review	23	28	91	6.2
Code	42	91	204	13.9
Code Review	16	47	89	6.1
Compile	0	11	12	0.8
Test	15	37	77	5.3
Postmortem	15	0	41	2.8
Total	274	725	1468	100.0
UPI (70%)	484			
LPI (70%)	64			

### PSP2.1 Project Plan Summary - Program 3A (Continued)

Student 94419-JUAN ALBERTO GUTIERREZ Date 20160601  
Program Chi-square Test Program# 3A  
Instructor JORGE RAFAEL AGUILAR CISI Language C

	Plan	Actual	To Date	To Date%
<b>Defects Injected</b>				
Planning	0.5	0	1	3.0
Design	8.5	3	21	63.6
Design Review	0.0	0	0	0.0
Code	2.8	5	11	33.3
Code Review	0.0	0	0	0.0
Compile	0.0	0	0	0.0
Test	0.0	0	0	0.0
Total Development	11.8	8	33	100.0

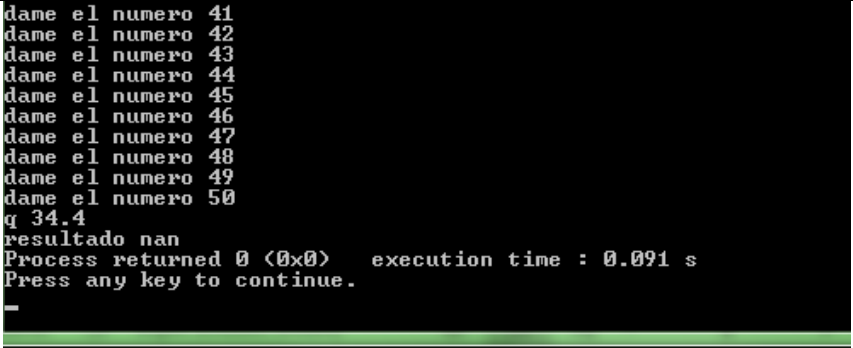
<b>Defects Removed</b>				
Planning	0.0	0	0	0.0
Design	0.5	1	2	6.1
Design Review	6.2	1	14	42.4
Code	2.4	1	6	18.2
Code Review	1.3	3	7	21.2
Compile	0.0	1	1	3.0
Test	0.3	1	3	9.1
Total Development	11.8	8	33	100.0
After Development		4	33	

<b>Defect Removal Efficiency</b>			
Def/Hr - DLDR	15.90	2.16	3.26
Def/Hr - CDR	7.31	3.83	4.71
Def/Hr - Compile	0.00	5.43	4.95
Def/Hr - Test	3.82	1.62	2.33
DRL(DLDR/UT)	4.16	1.34	3.98
DRL(CDR/UT)	1.91	2.37	2.02
DRL(Compile/UT)	0.00	3.36	2.13



### Test Report template

<b>Test Name/#</b>	1
<b>Test Objective</b>	Normalizar los datos
<b>Test Description</b>	Normalizar una lista de datos con la función para $x^2$
<b>Test Conditions</b>	50 5.67 5.75 5.80 6 6 6 7 7.33 7.5 7.57 7.67 7.80 8.33 8.33 8.67 8.67 13.75 14 14.5 14.67 16.4 16.4 19.2 19.33 20.5 21.75 22.25 24.17 25.42 28.33 29 29.67 8.67 8.83 9

	10 10.33 10.67 11.75 12 13.25 30.14 36.8 37.33 38 39 40.25 41 50.63 52.80
<b>Expected Results</b>	Q=34.4 1-p=7.60*10 <sup>-5</sup>
<b>Actual Results</b>	 <pre> dame el numero 41 dame el numero 42 dame el numero 43 dame el numero 44 dame el numero 45 dame el numero 46 dame el numero 47 dame el numero 48 dame el numero 49 dame el numero 50 q 34.4 resultado nan Process returned 0 (0x0)   execution time : 0.091 s Press any key to continue. </pre>

<b>Test Name/#</b>	2
<b>Test Objective</b>	Normalizar los datos
<b>Test Description</b>	Normalizar una lista de datos con la función para x <sup>2</sup>
<b>Test Conditions</b>	50 5.67 5.75 5.80 6 6 6 7 7.33



	7.5 7.57 7.67 7.80 8.33 8.33 8.67 8.67 13.75 14 14.5 14.67 16.4 16.4 19.2 19.33 20.5 21.75 22.25 24.17 25.42 28.33 29 29.67 8.67 8.83 9 10 10.33 10.67 11.75 12 13.25 30.14 36.8 37.33 38 39 40.25 41 50.63 52.80
<b>Expected Results</b>	Q=34.4 $1-p=7.60 \cdot 10^{-5}$



**Actual  
Results**

```
dame el numero 45
dame el numero 46
dame el numero 47
dame el numero 48
dame el numero 49
dame el numero 50
q 34.4
resultado 0.00301576
Process returned 0 (0x0)   execution time : 0.057 s
Press any key to continue.
```

**Los documentos de:**

**Estándar de codificación y estándar de conteo se encuentran en la carpeta de Codificación.**

**PSP2 Design Review checklist**

**PSP2 Design Review Checklist**

Student	Juan Alberto Gutierrez Canto	Date	15/07/2016
Program	Normalizacion	Program #	9A
Instructor	Jorge Rafael Aguilar Cisneros	Language	C++

<b>Purpose</b>	To guide you in conducting an effective design review
<b>General</b>	<ul style="list-style-type: none"><li>- Review the entire program for each checklist category; do not attempt to review for more than one category at a time!</li><li>- As you complete each review step, check off that item in the box at the right.</li><li>- Complete the checklist for one program or program unit before reviewing the next.</li></ul>

Complete	Verify that the design covers all of the applicable requirements. <ul style="list-style-type: none"><li>- All specified outputs are produced.</li><li>- All needed inputs are furnished.</li><li>- All required includes are stated.</li></ul>	MB X X X	B	R	M
External Limits	Where the design assumes or relies upon external limits, determine if behavior is correct at nominal values, at limits, and beyond limits.				
Logic	<ul style="list-style-type: none"><li>- Verify that program sequencing is proper. Stacks, lists, and so on are in the proper order. Recursion unwinds properly.</li><li>- Verify that all loops are properly initiated, incremented, and terminated.</li><li>- Examine each conditional statement and verify all cases.</li></ul>	X  X	X		
Internal Limits	Where the design assumes or relies upon internal limits, determine if behavior is correct at nominal values, at limits, and beyond limits.				
Special Cases	<ul style="list-style-type: none"><li>- Check all special cases.</li></ul>	X			





## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

	<ul style="list-style-type: none"><li>- Ensure proper operation with empty, full, minimum, maximum, negative, and zero values for all variables.</li><li>- Protect against out-of-limits, overflow, and underflow conditions.</li><li>- Ensure “impossible” conditions are absolutely impossible.</li><li>- Handle all possible incorrect or error conditions.</li></ul>	X			
		X			
		X			
		X			
Functional Use	<ul style="list-style-type: none"><li>- Verify that all functions, procedures, or methods are fully understood and properly used.</li><li>- Verify that all externally referenced abstractions are precisely defined.</li></ul>	X			
		X			
System Considerations	<ul style="list-style-type: none"><li>- Verify that the program does not cause system limits to be exceeded.</li><li>- Verify that all security-sensitive data are from trusted sources.</li><li>- Verify that all safety conditions conform to the safety specifications.</li></ul>	X			
		X			
		X			
Names	Verify that <ul style="list-style-type: none"><li>- all special names are clear, defined, and authenticated</li><li>- the scopes of all variables and parameters are self-evident or defined</li><li>- all named items are used within their declared scopes</li></ul>		X		
			X		
			X		
Standards	Ensure that the design conforms to all applicable design standards.	X			

### Code Review checklist

#### Code Review Checklist

Student	Juan Alberto Gutierrez Canto	Date	16/07/2016
Program	List sort	Program #	8A
Instructor	Jorge Rafael Aguilar Cisneros	Language	C++

<b>Purpose</b>	To guide you in conducting an effective code review
<b>General</b>	<ul style="list-style-type: none"><li>- Review the entire program for each checklist category; do not attempt to review for more than one category at a time!</li><li>- As you complete each review step, check off that item in the box at the right.</li><li>- Complete the checklist for one program or program unit before reviewing the next.</li></ul>

Complete	Verify that the code covers all of the design.	MB	B	R	M
Includes	Verify that the includes are complete.	X			
Initialization	Check variable and parameter initialization. <ul style="list-style-type: none"><li>- at program initiation</li><li>- at start of every loop</li><li>- at class/function/procedure entry</li></ul>	X			
		X			
		X			
Calls	Check function call formats.				



# PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

	<ul style="list-style-type: none"> <li>- pointers</li> <li>- parameters</li> <li>- use of '&amp;'</li> </ul>	X X X			
Names	Check name spelling and use. <ul style="list-style-type: none"> <li>- Is it consistent?</li> <li>- Is it within the declared scope?</li> <li>- Do all structures and classes use '.' reference?</li> </ul>		X X X		
Strings	Check that all strings are <ul style="list-style-type: none"> <li>- identified by pointers</li> <li>- terminated by NULL</li> </ul>	X X			
Pointers	Check that all <ul style="list-style-type: none"> <li>- pointers are initialized NULL</li> <li>- pointers are deleted only after new</li> <li>- new pointers are always deleted after use</li> </ul>	X X X			
Output Format	Check the output format. <ul style="list-style-type: none"> <li>- Line stepping is proper.</li> <li>- Spacing is proper.</li> </ul>	X X			
() Pairs	Ensure that () are proper and matched.				
Logic Operators	<ul style="list-style-type: none"> <li>- Verify the proper use of ==, =,   , and so on.</li> <li>- Check every logic function for ().</li> </ul>	X X			
Line-by-line check	Check every line of code for <ul style="list-style-type: none"> <li>- instruction syntax</li> <li>- proper punctuation</li> </ul>			X X	
Standards	Ensure that the code conforms to the coding standards.				
File Open and Close	Verify that all files are <ul style="list-style-type: none"> <li>- properly declared</li> <li>- opened</li> <li>- closed</li> </ul>	X		X X	

**PIP form**

Programa 2 A		
No. Del PIP	Descripción del Problema	Descripción de la Propuesta
1	No tenía todos los documentos y materiales que se necesitaban a la mano.	Preparar todos los materiales antes de empezar a trabajar.
2	Falto hacer mejor el diseño y los requerimientos	Tener unos templates de requerimientos y de diseño
3	Todo el trabajo se hizo bajo presión por poco tiempo	Planear la creación del proyecto con anterioridad
Programa 3 A		
No. Del PIP	Descripción del Problema	Descripción de la Propuesta
4	Muchas distracciones por el ambiente de trabajo	Trabajar en un lugar donde haya menos ruido
5	Falta añadir un mejor diseño	Utilizar el diagrama de clases, diagrama de casos de uso, diagrama de actividades y diagrama de estados.
Programa 4 A		
No. Del PIP	Descripción del Problema	Descripción de la Propuesta
6	Todo el trabajo se realizo bajo la presión del tiempo	Realizar el trabajo con anticipación
7	Hubo conceptos que no se entendían	Ir a las asesorías o enviar correo para preguntar sobre el proceso
Programa 5 A		
No. Del PIP	Descripción del Problema	Descripción de la Propuesta
8	No se tomo en cuenta el nivel del psp	Preguntar a el tutor por el nivel antes de empezar a realizar el programa
Programa 6 A		
No. Del PIP	Descripción del Problema	Descripción de la Propuesta
Programa 7 A		
No. Del PIP	Descripción del Problema	Descripción de la Propuesta
9	No se entendio complementamente el problema a resolver	Tratar de poner mas atención y definir mejor el problema
10	Especificar mejor el diseño	Utilizar mejores procesos de PSP



# PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

## Programa 8 A

No. Del PIP	Descripción del Problema	Descripción de la Propuesta
11	No se sigue el calendario de planeación para el trabajo	Al momento de planear los días que se trabajarán, tener en cuenta las actividades fuera de la clase.

## Programa 9 A

No. Del PIP	Descripción del Problema	Descripción de la Propuesta
12	No se entienden las funciones anteriormente echas	Hacer funciones para el reuso.

## SIZE ESTIMATING TEMPLATE

### Size Estimating Template

Student  
Instructor

JUAN ALBERTO GUTIERREZ C, Date  
JORGE RAFAEL AGUILAR CIS, Program#

#####  
9A

#### BASE PROGRAM LOC

BASE SIZE (B) =>

LOC DELETED (D) =>

LOC MODIFIED (M) =>

ESTIMATE

ACTUAL

~~105~~

~~95~~

~~30~~

~~20~~

~~50~~

~~15~~

#### OBJECT LOC

BASE ADDITIONS:

PEDIRDATOSO

TYPE	METHODS	REL. SIZE
data	1	S

LOC	LOC
0.0	15

(BA) subtotal from page 2

TOTAL BASE ADDITIONS (BA)

0.0

0

0.0

15

NEW OBJECTS:

REGLASMPNOR()
REGLASPMX2()
NORMALIZACION()
AVG()
VARIANZA
SECNORMAL()
PROBABILIDAD()

TYPE	METHODS	REL. SIZE
Logic	1	S
Logic	1	VL
Logic	1	VL
calc	1	s
Calc	1	s
Logic	1	s
Calc	1	s

LOC *	LOC *
11.0	50
33.8	56
33.8	26
0.0	14
0.0	14
0.0	53
0.0	4

(NO) subtotal from page 2

TOTAL NEW OBJECTS (NO)

0.0

0

78.6

219



# PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

TOTAL NEW OBJECTS (NO)	18.6	219
REUSED OBJECTS	LOC	LOC
CALCULARRANGO()	6.0	0
XI()	9.0	0
XI2()	4.0	0
EXPX()	4.0	0
FXI()	4.0	0
TERM()	13.0	0
RESULTADO()	10.0	0
(R) subtotal from page 2	0.0	0
REUSED TOTAL (R)	50.0	0

		Size	Time
Estimated Object LOC:	$E = BA + NO + M$	128.64	
Regression Parameter:	$B_0$	22.57	27.55
Regression Parameter:	$B_1$	0.88	1.92
Estimated New and Changed LOC:	$N = B_0 + B_1 * E$	135.5	
Estimated Total LOC:	$T = N + B - D - M + R$	210.5	
Estimated Total New Reuse (sum of * LOC):		0	
Estimated Total Development Time:	$Time = B_0 + B_1 * E$		274.5
Prediction Range:	Range	50.5	210.0
Upper Prediction Interval:	$UPI = N + Range$	186.0	484.5
Lower Prediction Interval:	$LPI = N - Range$	85.0	64.5
Prediction Interval Percent		70%	70%
Method Selected		A	A
R <sup>2</sup>		0.67	0.36

### Size Estimating Template (continued)

Student  
Instructor

JUAN ALBERTO GUTIERREZ CAN Data  
JORGE RAFAEL AGUILAR CISNEI Program#

07/07/2016  
9A

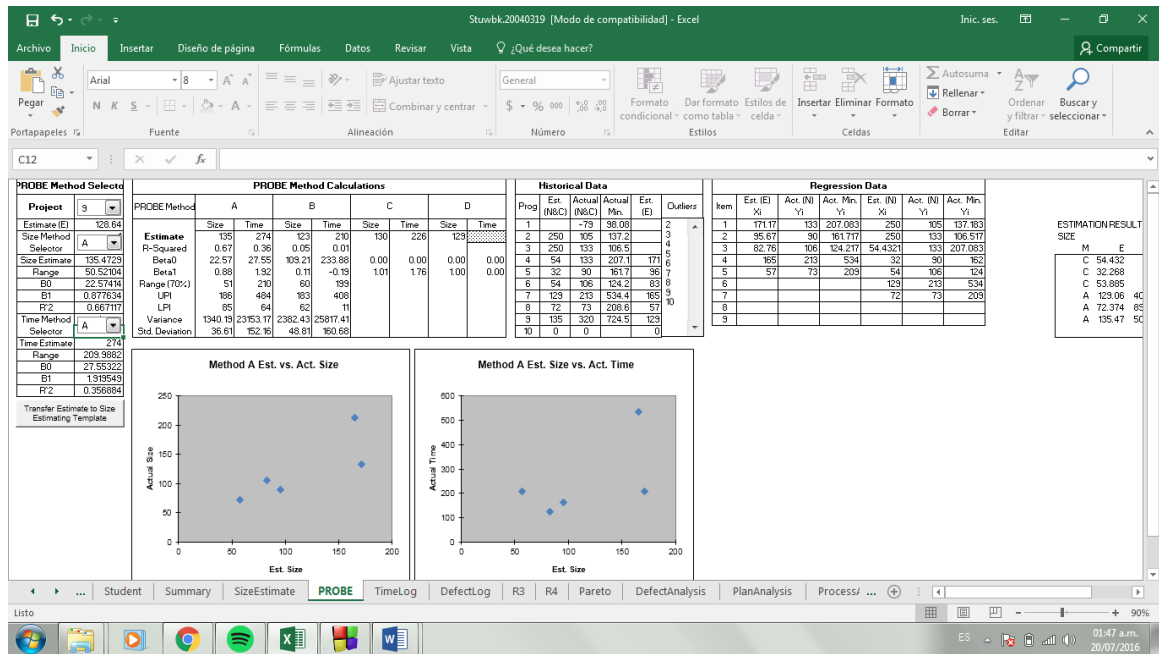
[illegible]



## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

### PROBE Calculation worksheet



### Time Recording Log.



## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

	A	B	C	D	E	F	G	H
1	Project	Phase	Date	Start	Int.	Stop	Delta	Comments
69	8	DLD	06/25/16	12:26:38		12:52:07	25.5	Revisión contemplando el estándar
70	8	CODE	06/25/16	12:52:18		13:10:41	18.4	Solamente se paso del diseño al código, osea muy facil
71	8	CR	06/26/16	12:01:11		12:18:02	16.8	Se revizo conforme al estándar
72	8	COMPILE	06/26/16	12:18:12		12:18:13	0.0	Sin errores la compilación
73	8	TEST	06/26/16	12:18:21		12:22:38	4.3	Sin errores las pruebas
74	8	PM	06/26/16	12:23:00		12:33:46	10.8	Completando el proceso
75	9	PLAN	07/07/16	20:19:51		20:34:32	14.7	
76	9	PLAN	07/08/16	19:35:01		20:05:19	30.3	
77	9	PLAN	07/09/16	10:35:54		12:12:58	97.1	
78	9	PLAN	07/11/16	20:40:31		21:10:42	30.2	
79	9	DLD	07/12/16	22:41:05		00:18:03	97.0	
80	9	DLD	07/13/16	20:18:17		21:07:53	49.6	
81	9	DLD	07/14/16	14:49:36		16:31:17	101.7	
82	9	DLD	07/15/16	00:33:15		02:03:41	90.4	
83	9	DLD	07/15/16	15:34:21		16:02:09	27.8	
84	9	CODE	07/15/16	22:03:54		23:34:31	90.6	
85	9	CR	07/16/16	11:27:09		12:14:12	47.0	
86	9	COMPILE	07/16/16	12:35:13		12:46:16	11.1	
87	9	TEST	07/16/16	12:49:40		13:26:47	37.1	
88		PM	07/19/16	13:07:41		14:21:11	73.5	
89								
90								
91								
92								
93								
94								
95								

### Operational Specification Template

### Operational Specification Template

<b>Student</b>	Juan Alberto Gutierrez Canto	<b>Date</b>	12/07/2016
<b>Program</b>	Normalization	<b>Program #</b>	9A
<b>Instructor</b>	Jorge Rafael Aguilar Cisneros	<b>Language</b>	C++

Scenario Number	1	User Objective	Ver la calidad de un grupo de datos
Scenario Objective	Pasos para la normalización de datos.		
Source	Step	Action	Comments
Usuario	1	Abrir el program.	
Programa	2	Pedir número de datos que va a introducir.	El numero n de datos.
Usuario	3	Ingresar el número de datos	Recibe n.
Programa	4	Pedir los datos x,y.	De 0 a n.
Usuario	5	Ingresa los datos x, y.	Insertarlos en la lista.
Programa	6	Guardar datos en lista	Guarda los datos en lista ligada
Programa	7	Ordenar los datos.	Ordenarlos en la lista de forma ascendente.
Programa	8	Calcular promedio de datos.	
Programa	9	Calcular varianza	
Programa	10	Calcular desviación estándar	





<b>Student</b>	Juan Alberto Gutierrez Canto	<b>Date</b>	13/07/2016
<b>Program</b>	Normalizacion	<b>Program #</b>	9A
<b>Instructor</b>	Jorge Rafael Aguilar Cisneros	<b>Language</b>	C++



## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

<b>Class Name</b>	Int ISEMPTY()
<b>Parent Class</b>	N/A

Attributes	
Declaration	Description
Struc nodo Raiz	El nodo de el comienzo o el raiz.

Items	
Declaration	Description
Return	Regresa un valor declarado.

<b>Class Name</b>	Struct nodo
<b>Parent Class</b>	N/A

Attributes	
Declaration	Description
Double numerox;	Numero de cada nodo llamado x.
Double numeroy;	Numero de cada nodo llamado y.
Struct nodo sig;	Apuntador a la siguiente lista;

Items	
Declaration	Description
N/A	

<b>Class Name</b>	Void INSERT()
<b>Parent Class</b>	N/A

Attributes	
Declaration	Description
Struct nodo nuevo;	Nodo para nuevos datos;
Double datox;	Dato x, mandado del usuario.
Double datoy;	Dato y, mandado del usuario.

Items	
Declaration	Description
ISEMPTY()	Para revisar si la lista esta vacía.
Malloc()	Genera espacio en memoria.
Sizeof()	Regresa el tamaño de un dato.



## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

<b>Class Name</b>	Void REMUEVE()
<b>Parent Class</b>	N/A

Attributes	
Declaration	Description
Struct nodo *elimina;	Nodo a eliminar de la lista.

Items	
Declaration	Description
ISEMPTY()	Regresa si la lista esta vacía =1, o si no =0.
Free()	Limpia un espacio en memoria.

<b>Class Name</b>	Void PEDIRDATOS()
<b>Parent Class</b>	N/A

Attributes	
Declaration	Description
Doublé datox;	Dato x, introducido por el usuario.
Double datoy;	Dato y, introducido por el usuario.
Long Long n;	Numero de datos que introducirá el usuario.
Long Long i;	Contador de datos.

Items	
Declaration	Description
INSERT()	Inserta los datos en la lista ligada.

<b>Class Name</b>	Void BURBUJALTST()
<b>Parent Class</b>	N/A

Attributes	
Declaration	Description
Struct nodo *inicio;	Nodo para el comienzo de la lista.
Struct nodo *siguiente;	Nodo para comparar la lista.
Doublé datox;	Dato auxiliar para hacer el cambio de elementos en x.
Long i;	Contador de números dentro de la lista.
Long j;	Contador de números dentro de la lista.
Doublé datoy;	Dato auxiliar para hacer el cambio de elementos en y.

Items	
Declaration	Description



## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

N/A

<b>Class Name</b>	double NORMALIZACION()
<b>Parent Class</b>	N/A

Attributes	
Declaration	Description
double xavg	Variable para el average de los datos x.
double varianza	Variable para almacenar la varianza.
double dest	Variable para almacenar la desviación estandar de la función.
double Q	Calidad de los datos normalizados.
double simx2	Regla de Simpson con $x^2$
double p	Probabilidad tail.
Nodo inicio	Tabla de valores

Items	
Declaration	Description
AVG()	Saca el average de una serie de datos.
VARIANZA()	Calcula la varianza de una serie de datos.
SECNORMAL()	Calcula los segmentos y la calidad de los datos.
REGLASMPX2()	Regal de Simpson con $x^2$
PROBABILIDAD()	Probilidad de una serie de datos.
Sqrt()	Calcula la raíz cuadrada de una serie de datos.
Return	Regresa un dato de tipo float.

<b>Class Name</b>	double AVG
<b>Parent Class</b>	N/A

Attributes	
Declaration	Description
Struct nodo *inicio;	Nodo para el comienzo de la lista.
Double avgdata;	Contador para el dato.
long i	Utilizado para los ciclos for

Items	
Declaration	Description
Return	Regresa un dato en especifico



## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

<b>Class Name</b>	double VARIANZA
<b>Parent Class</b>	N/A

Attributes	
Declaration	Description
Struct nodo *inicio;	Nodo para el comienzo de la lista.
Double varianza;	Contador para el dato, y formulas.
long i	Utilizado para los ciclos for
Double avgdata	Utilizado para el average

Items	
Declaration	Description
Return	Regresa un dato en especifico
Pow	Para elevar un termino a un exponente dado

<b>Class Name</b>	double SECNORMAL()
<b>Parent Class</b>	N/A

Attributes	
Declaration	Description
Struct nodo *inicio;	Nodo para el comienzo de la lista.
Double tablavaloresnor	Valores para hacer la tabla de valores.
Double sumterm.	Suma de valores de los términos.
Double rglsmps	Valores para los rangos comensando en -4 hasta 4
long segmentos	Numero de segmentos

Items	
Declaration	Description
Return	Regresa un dato en especifico
CALRANGOS()	Calcula los rangos para S segmentos
REGLASMPNOR()	Calcula la regla de Simpson con un valor determinado.

<b>Class Name</b>	int CALRANGOS()
<b>Parent Class</b>	N/A

Attributes	
------------	--



## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

Declaration	Description
Int n	Numero de datos.
Int s	Segmentos de la lista
Double tablavaloresnor	Se utiliza la tabla.
Int i	Se utiliza para los ciclos

Items	
Declaration	Description
Return	Regresa un dato en especifico

<b>Class Name</b>	Double REGLASMPNOR ()
<b>Parent Class</b>	N/A

Attributes	
Declaration	Description
Double xlow	Valor de x bajo
Double xhigh	Valor de x alto
Double N	Numero de bloques
Double W	Pesos de incremento.
Double fija	Variable fija de la función que corresponde a $(1/\sqrt{2 * \pi})$
Double x2	Valor para la ecuacion
Double ex	Valor de e para la ecuación
Double fx	Representa el valor de toda la formula
Double term	Termino de la función
Double sumterm	Sumatoria de términos
Bool negative	Utilizado para saber que variable entre si positiva o negativa
Int i	Utilizado para los ciclos

Items	
Declaration	Description
Return	Regresa un dato en especifico
Sqrt()	Regresa la raíz cuadrada de un determinado numero.
Pow()	Regresa el cuadrado de un numero.
Exp()	Regresa el valor de e a un exponente dado.

<b>Class Name</b>	Double REGLASPMX2()
-------------------	---------------------



## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

Parent Class	N/A
--------------	-----

### Attributes

Declaration	Description
Double xlow	Valor de x bajo
Double xhigh	Valor de x alto
Double N	Numero de bloques
Double n	Grados de libertad
Double tfija	Valor de t fijo con la condición de $\tau(\frac{n}{2})$
Double fijo	Valor fijo de $2^{n/2}$
Double xan	Valor de $x^{n/2-1}$
Double exa	Valor de e a el exponente $-(x/2)$
Double fx	Valor de la ecuacion
Double term	Valor del termino de la ecuacion
Bool negative	Valor para saber si es negativo o positivo xhigh
Int i	Utilizado para los ciclos
Double sumter	Valor que suma los términos

### Items

Declaration	Description
Return	Regresa un dato en especifico
Pow()	Eleva un valor a el exponente dado
Exp((	Eleva la variable e a un exponente determinado.

Class Name	double PROBABILIDAD()
------------	-----------------------

Parent Class	N/A
--------------	-----

### Attributes

Declaration	Description
Double prob	Data para calcular la probabilidad

### Items

Declaration	Description
Return	Regresa un dato en especifico



## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

<b>Class Name</b>	int main()
<b>Parent Class</b>	N/A

Attributes	
Declaration	Description

Items	
Declaration	Description
Freeopen()	Leer un archivo de texto.
PEDIRDATOS()	Pedir los datos al usuario.
BURBUJALST()	Ordenar la lista.
NORMALIZACION()	Hace la normalización de los datos ingresados.
Printf()	Imprime datos a pantalla.

## State Specification Template

### State Specification Template

Student	Juan Alberto Gutierrez Canto	Date	14/06/2016
Program	Normalizacion	Program #	9A
Instructor	Jorge Rafael Aguilar Cisneros	Language	C++

#### ORDENAMIENTO

State Name	Description	
Inicio	Inicia el programa.	
Pedir n datos	El usuario debera de escribir cuantos datos necesita.	
Pedir x,y datos	El usuario ingresa datos x, y.	
Guardar en lista	Se guardan x, y datos en un nodo de la lista	
Lista comenzada	Inserta datos desde el ultimo lugar de la lista.	
Lista vacia	Inserta dato en una lista desde raíz.	
Ordenar lista	Se ordena la lista ligada	
Fin	Termina el programa	
Function/Parameter	Description	
INSERT()	Inserta datos en la lista ligada.	
ISEMPTY()	Verifica que la lista esta vacia.	
N	Numero de datos de la lista.	
I	Contador de números que a ingresado.	
BURBUJALTS()	Función para ordenar datos.	
Ambas	Para checar que se ordeno ambos datos	
States/Next States	Transition Condition	Action
Inicio		
Pedir n datos	True	Get n;
Pedir n datos		





Pedir x,y datos	N>0;	Get x, get y; i=0;
Pedir x,y datos		
Guardar en lista	INSERT(x,y)	
Pedir x,y datos	i<n	I++; get x, get y;
Ordenar lista	i>n	BURBUJALTS();ambas=false
Guardar en lista		
Lista vacia	ISEMPTY()=yes	Raíz+1;
Lista comenzada	ISEMPTY()=no	Last+;
Lista vacia		
Pedir x,y datos	True	Get x, get y;
Lista comenzada		
Pedir x,y datos	True	Get x, get y;
Ordenar lista		
Ordenar lista	Ambas=false	BURBUJALTS();ambas=true, PRINTF()
Fin	Ambas=true	PRINTF();
Fin		
Fin	True	

## REGLA DE SIMPSON NORMAL



# PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

State Name	Description	
Xi	Termino de la ecuacion.	
Xi2	Termino al cuadrado	
EX	Termino con exponencial	
FX	Termino en la formula	
TERM	Termino si es primero o ultimo o par e impar.	
Xlow	X con el valor mas bajo	
Xhigh	X con el valor mas alto	
N	Numero de bloques	
W	Pesos para la ecuación	
fijo	Valor fijo de la ecuación	
States/Next States	Transition Condition	Action
XI		
Xi2	true	Xlow+w;
Xi2		
EX	True	Xlowi^2/2
EX		
FX	True	E^-xi^2/2
FX		
TERM	N ==0    N==20, N!= par, N!=impar	FX*w/3, 2*FX*w/3, 4*FX*w/3
TERM		
XI	N!=20	N++
Fin	N==20	

## REGLA DE SIMPSON ^2

State Name	Description	
Xi	Termino de la ecuacion.	
Xi2	Termino al cuadrado	
EX	Termino con exponencial	
FX	Termino en la formula	
TERM	Termino si es primero o ultimo o par e impar.	
Xlow	X con el valor mas bajo	
Xhigh	X con el valor mas alto	
N	Numero de bloques	
n	Numero de grados de libertad	
T	Valor de t	
W	Pesos para la ecuación	
fijo	Valor fijo de la ecuación	
States/Next States	Transition Condition	Action
XI		
Xi2	true	Xlow+w;
Xi2		
EX	True	Xlowi^2/2
EX		
FX	True	E^-xi^2/2
FX		
TERM	N ==0    N==20, N!= par, N!=impar	FX*w/3, 2*FX*w/3, 4*FX*w/3
TERM		



## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

XI	N!=20	N++
Fin	N==20	

### Logic Specification Template

#### Logic Specification Template

Student	Juan Alberto Gutierrez Canto	Date	14/06/2016
Program	Normalizacion	Program #	9A
Instructor	Jorge Rafael Aguilar Cisneros	Language	C++
Object	N/A	Function	ISEMPTY()

#### INCLUDES:

**TYPE DEFINITIONS:** Using namespace std;

#### Declaration:

#### Reference:

Logic reference numbers	Program logic, in pseudocode
1	Si el nodo llamado raiz es igual a null
	En caso que si regresar 1
	En caso que no regresar 0



## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

Object N/A Function Struct nodo

### INCLUDES:

**TYPE DEFINITIONS:** Using namespace std;

### Declaration:

Double numero x

Double numeroy

Struct nodo\*sig

Struct nodo raiz=NULL

Struct nodo last=NULL

### Reference:

Logic reference numbers	Program logic, in pseudocode
1	Declaración de variables
	El nodo llamado raíz es null
	El nodo llamado last es null



## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

Object N/A Function INSERT()

**INCLUDES:** `#include <malloc.h>`  
`#include <studio.h>`

**TYPE DEFINITIONS:** Using namespace std;

**Declaration:** Struct nodo \*Nuevo=NULL

**Reference:** \_\_\_\_\_

Logic reference numbers	Program logic, in pseudocode
1	Si la lista esta vacia
	En caso que si
	Reservamos memoria para el nodo
	Guardamos valor de x
	Guardamos valor de y
	Apuntamos siguiente como nulo
	Raíz es igual a last que es igual a nuevo
	En caso de que no
	Reservamos memoria para el nodo
	Guardamos valor de x
	Guardamos valor de y
	Apuntamos siguiente como nulo
	Apuntamos sig de last como nuevo
	Last es igual a nuevo



## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

Object N/A Function REMUEVE

**INCLUDES:** `#include <stdio.h>`  
`#include <stdlib.h>`

**TYPE DEFINITIONS:** Using namespace std;

**Declaration:** Struct nodo \*elimina=NULL

**Reference:** \_\_\_\_\_

Logic reference numbers	Program logic, in pseudocode
1	Si la lista no esta vacia
	En caso de que si
	Elimina es igual a raíz
	Raíz es igual a raíz que apunta en siguiente
	Liberamos espacio de memoria de elimina



## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

Object N/A Function PEDIRDATOS

**INCLUDES:** `#include <stdio.h>`

**TYPE DEFINITIONS:** Using namespace std;

**Declaration:**

Double datox

Double datoy

Long long i

**Reference:**

Logic reference numbers	Program logic, in pseudocode
1	Pedir n datos
	Leer n datos
2	Inicializar i=0
	Mientras i<n
	Pedir x & y
	Leer x & y
	Insertar en la lista ligada
	I++



## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

Object N/A Function BURBUJALTST

**INCLUDES:** #include <stdio.h>

**TYPE DEFINITIONS:** Using namespace std;

**Declaration:**

```
Struct nodo *inicio
Struct nodo *siguiente
Double datox
Long long i
Long long y
Bool ambas=false
Double datoy
```

**Reference:**

Logic reference numbers	Program logic, in pseudocode
1	Inicialiar i=0
	Inicializar y=0
	Inicio es igual a raíz
2	Mientras i<n
	Inicializar y=i+1
3	Auxiliar es igual a lo que apunta incio en siguiente
	Mientras y<n
4	Si ambas es falsa
	En caso que si
5	Si datox de incio es mayor a datox de Auxiliar
	En caso que si
	Datox es igual a lo que hay en datox de Auxiliar
	Datoy es igual a lo que hay en datoy de Auxiliar
	Datox de Auxiliar es igual a lo que hay en datox de incio
	Datoy de Auxiliar es igual a lo que hay en datoy de incio
	Datox de incio es igual a datox
	Datoy de incio es igual a datoy
	En caso de que no
	Si datoy de incio es mayor a datoy de siguiente
	En caso que si
	Datox es igual a lo que hay en datox de Auxiliar
	Datoy es igual a lo que hay en datoy de Auxiliar
	Datox de Auxiliar es igual a lo que hay en datox de incio
	Datoy de Auxiliar es igual a lo que hay en datoy de incio
	Datox de incio es igual a datox
	Datoy de incio es igual a datoy
	Y++
	Auxiliar es igual a lo que apunta Auxiliar en siguiente
	I++
	Inicio es igual a lo que apunta incio en siguiente





PROCESO PERSONAL DE DESARROLLO DE  
SOFTWARE 2

VERANO 2016

	Inicialiar $i=0$
	Inicio es igual a raíz
6	Mientras $i < n$
	Imprime el valor de $x$ & $y$
	$I++$
	Inicio es igual a lo que apunta inicio en siguiente
7	Si ambas es igual a false
	En caso de que si
	Ambas es igual a true
	BURBUJALTST()



## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

Object N/A Function AVG

### INCLUDES:

**TYPE DEFINITIONS:** Using namespace std;

### Declaration:

Struct nodo \*inicio

Double avgdata

Int i

### Reference:

Logic reference numbers	Program logic, in pseudocode
	Inicio = raíz;
	Avgdata=0;
1	Para i=0, mientras i<n, i++
	Avgdata+=numero x en inicio
	Incio=sig en inicio;
	Avgdata=avgdata/n
	Return avgdata



## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

Object N/A Function VARIANZA

**INCLUDES:** `#include <math.h>`

**TYPE DEFINITIONS:** `Using namespace std;`

**Declaration:**

`Struct nodo *inicio`

`Double varianza`

`Int i`

`Double avgdata`

**Reference:**

Logic reference numbers	Program logic, in pseudocode
	<code>Inicio = raíz;</code>
	<code>Varianza=0;</code>
1	<code>Para i=0, mientras i&lt;n, i++</code>
	<code>Varianza+=pow(Numero en inicio -avgdata,2)</code>
	<code>Inicio=inicio-&gt;sig;</code>
	<code>varianza=varianza/n-1</code>
	<code>Return varianza</code>



## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

Object N/A Function CALRANGOS

### INCLUDES:

**TYPE DEFINITIONS:** Using namespace std;

### Declaration:

Int n  
Int s  
Double tablavaloresnor  
Int i

### Reference:

Logic reference numbers	Program logic, in pseudocode
1	If(n mod 5 == 0)
2	Para i=0, mientras i< n/5, i++ Tablavaloresnor en posición 3,i=5;
3	Else if(n mod 5 == 4)
4	Para i=0,mientras i<n/5, i++ Tablavaloresnor en pocicion 3,i=5 Tablavaloresnor en pocicion 3,i=4
5	Else if(n mod 5<4)
6	Para i=0,mientras i<n/5-1,i++ Tablavaloresnor en pocicion 3,i = 5
7	Para i=0 mientras i< n mod 5 + 5,i++ Tablavaloresnor en posición 3, i+1 ++
8	Return n/5-1



Object N/A Function REGLASMPNOR

**INCLUDES:** `#include <math.h>`

**TYPE DEFINITIONS:** Using namespace std;

**Declaration:**

Double xlow
Double xhigh
Double N
Double W
Double fija
Double x2
Double ex
Double fx
Double term
Double sumterm
Bool negative

**Reference:**

Logic reference numbers	Program logic, in pseudocode
	Negative=false
	Fija=1/sqrt(2*pi)
	N=20
1	If(xhigh <0)
	Xhigh =xhigh*-1
	Negative=true
	W= xhigh/N
2	Para i=0, mientras i<=20, de i++
	X2=pow(xlow,2)/2
	Ex=exp(-x2)
	Fx=ex/fija
3	If(i==0    i==20)
	Term=fx*w/3
4	Else If(imod2 ==0)
	Term=2*fx*w/3
5	else



	Term=4*fx*w/3
	Sumterm+=term
	Xlow+=w;
6	If(negative)
	Sumterm-=0.5
7	Else
	Sumterm+=0.5
8	Return sumterm

Object N/AFunction SECNORMAL**INCLUDES:****TYPE DEFINITIONS:** Using namespace std;**Declaration:**Struc nodo inicioDouble tablavaloresnorDouble sumtermDouble rglsmplsInt segmentosdouble pkDouble contsegmentDouble paboveDouble pbelowDouble xbelowDouble xaboveDouble busquedaDouble contador**Reference:**

Logic reference numbers	Program logic, in pseudocode
	Segmentos=CALRANGOS()
	Rglsmpls=-4
	Sumterm=0
	Contsegment=0
	valregla=0
	Inicio =raíz;
	Búsqueda = numeroy de inicio
	Tablavaloresnor en posición 1,0 = -100000
	Para i=0, mientras i<segmentos, de i++



PROCESO PERSONAL DE DESARROLLO DE  
SOFTWARE 2

VERANO 2016

	Contsegment+=tablavaloresnor en posición 3, i
	pk=1/contsegment;
	Mientras pabove< pk
	Pbelow=pabove
	Xbelow=xabove
	pabove=REGLASMPNOR(rglsmps)
	Xabove=rglsmps
	Rglsmpls=-.1
	Tablavaloresnor en posición 2,i= xbelow+((pk-pbelow)/(pabove-pbelow))*(xabove-xbelow)
	Tablavaloresnor en posición 1,i+1= xbelow+((pk-pbelow)/(pabove-pbelow))*(xabove-xbelow)
	Contador=0
	Mientras busqueda<tablavalores en posicion 2,i
	Contador++
	Inicio = sig que esta en inicio
	Búsqueda= numeroy de inicio
	Tablavaloresnor en posición 4,i=contador
	Tablavaloresnor en posición 5,i=pow(tablavaloresnor 3,i – tablavaloresnor 4,i , 2)
	Tablavaloresnor en posición 6,i=tablavaloresnor 5,i / tablavaloresnor 3,i
	Sumaterm+=tablavaloresnor 6,i
	Return sumterm;



PROCESO PERSONAL DE DESARROLLO DE  
SOFTWARE 2

VERANO 2016

Object N/A Function REGLASPMx2

**INCLUDES:** `#include <math.h>`

**TYPE DEFINITIONS:** Using namespace std;

**Declaration:**

Double xlow
Double xhigh
Double N
Double n
Double W
Double tfija
Double fija
Double xan
Double exa
Double fx
Double term
Double sumter
Bool negative

**Reference:**

Logic reference numbers	Program logic, in pseudocode
	Negative=false





	n=9
	Tfija=1
	Para i=1, mientras i<n,i+=2
	Tfija=tfija*i/2
	Tfija=tfija*sqrt(PI)
	Fija=pow(2,n/2)
	N=20
1	If(xhigh <0)
	Xhigh =xhigh*-1
	Negative=true
	W= xhigh/20
2	Para i=0, mientras i<=20, de i++
	Xan=pow(xlow,(n/2)-1)
	Exa=exp(-xlow/2)
	Fx=(xan*exa)/(fija*tfija)
3	If(i==0    i==20)
	Term=fx*w/3
4	Else If(imod2 ==0)
	Term=2*fx*w/3
5	Else
	Term=4*fx*w/3
	Sumterm+=term
	Xlow+=w;
6	If(negative)
	Sumterm-=0.5
7	Else
	Sumterm+=0.5
8	Return sumterm



PROCESO PERSONAL DE DESARROLLO DE  
SOFTWARE 2

VERANO 2016

Object N/A Function PROBABILIDAD

**INCLUDES:** `#include <math.h>`

**TYPE DEFINITIONS:** Using namespace std;

**Declaration:**

Double prob

**Reference:**

Logic reference numbers	Program logic, in pseudocode
	Prob=1-prob
	Return prob



## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

Object N/A Function NORMALIZACION

**INCLUDES:** `#include <math.h>`

**TYPE DEFINITIONS:** Using namespace std;

**Declaration:**

Double xavg

Double varianza

Double dest

Double Q

Double simx2

Double p

Nodo inicio

Int i

**Reference:**



## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

Logic reference numbers	Program logic, in pseudocode
	Xavg=AVG()
	Varianza=VARIANZA(xavg)
	Varianza=sqrt(varianza)
	Para i=0,mientras i<n,de i++
	Numeroy de incio=numerox de inicio-xavg/varianza
	Inicio=sig que esta en inicio
	Q=SECNORMAL()
	P=REGLASPMx2(q)
	P=PROBABILIDAD(p)
	Return p

Object N/A Function main

**INCLUDES:** #include <stdio.h>

**TYPE DEFINITIONS:** Using namespace std;

**Declaration:** \_\_\_\_\_

**Reference:** \_\_\_\_\_

Logic reference numbers	Program logic, in pseudocode
1	Freopen()
2	PEDIRDATOS()
3	BURBUJALTST()



PROCESO PERSONAL DE DESARROLLO DE  
SOFTWARE 2

VERANO 2016

	Printf(NORMALIZACION())
	Reurn 0



## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

Object N/A Function BURBUJALTST

**INCLUDES:** #include <stdio.h>

**TYPE DEFINITIONS:** Using namespace std;

**Declaration:**

```
Struct nodo *inicio
Struct nodo *siguiente
Double datox
Long long i
Long long y
Bool ambas=false
Double datoy
```

**Reference:**

Logic reference numbers	Program logic, in pseudocode
1	Inicialiar i=0
	Inicializar y=0
	Inicio es igual a raíz
2	Mientras i<n
	Inicializar y=i+1
3	Auxiliar es igual a lo que apunta incio en siguiente
	Mientras y<n
4	Si ambas es falsa
	En caso que si
5	Si datox de incio es mayor a datox de Auxiliar
	En caso que si
	Datox es igual a lo que hay en datox de Auxiliar
	Datoy es igual a lo que hay en datoy de Auxiliar
	Datox de Auxiliar es igual a lo que hay en datox de incio
	Datoy de Auxiliar es igual a lo que hay en datoy de incio
	Datox de incio es igual a datox
	Datoy de incio es igual a datoy
	En caso de que no
	Si datoy de incio es mayor a datoy de siguiente
	En caso que si
	Datox es igual a lo que hay en datox de Auxiliar
	Datoy es igual a lo que hay en datoy de Auxiliar
	Datox de Auxiliar es igual a lo que hay en datox de incio
	Datoy de Auxiliar es igual a lo que hay en datoy de incio
	Datox de incio es igual a datox
	Datoy de incio es igual a datoy
	Y++
	Auxiliar es igual a lo que apunta Auxiliar en siguiente
	I++
	Inicio es igual a lo que apunta incio en siguiente



## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

	Inicialiar i=0
	Inicio es igual a raíz
6	Mientras i<n
	Imprime el valor de x & y

### Defect Recording Log.

1	Project	Date	Num	Type	Injected	Removed	FixTime	Fix Ref.	Description
59	7	#####	58	DLD	DLD	DLD	2.0		se cambiaron las variables r,t,p para que fueran globales, y por lo tanto se quito el traslado de ellas por las funciones
60	7	#####	59	DLD	CODE		0.5		en el diagram de estados en la funcion de correlacion era sum en vez de suma para cada termino
61	7	#####	60	DLD	CODE		1.0		la variable t de el diagrama de clases en la funcion significancia se me olvido borrarla
62	7	#####	61	DLD	CODE		1.0		la funcion CALCULART3 le sobrara una R en el diagrama de clases
63	7	#####	62	DLD	CODE		1.0		en el diagrama de estados en las res se debia de llamar val para las funciones CALCULART1,CALCULART2
64	7	#####	63	CODE	CR		3.0		en la linea 187 me faltaba un parentesis en la formula
65	7	#####	64	CODE	CR		1.0		en la funcion correlacion habia un return 0 de mas
66	7	#####	65	CODE	CR		1.0		faltaba un par de parentesis en la funcionEXPX en la formula del valor
67	7	#####	66	CODE	CR		1.0		faltaba agregar la funcion de archivo freopen en el main
68	7	#####	67	CODE	TEST		20.0		Cuando se hace el calculo de gama2 se debe de calcular la multiplicacion de i/2 donde i=n y se decrementa en 2, yo tenia sumatoria y no multiplicacion
69	7	#####	68	CODE	TEST		15.0		falta inicializar la regla de simpson por cada iteracion
70	8	#####	69	PLAN	DLD		1.0		Faltaba agregar la funcion struc a el size estimated template.
71	8	#####	70	DLD	DLD		2.0		Confundi el nombre de la variable Auxiliar con siguiente en el documento de diseño Logic Specification Template
72	8	#####	71	DLD	DLD		2.0		Me faltaba por la variable booleana llamada ambas en el diagram de clases
73	8	#####	72	DLD	CODE		2.0		Estaba haciendo un ciclo de mas a la hora de imprimir los datos ordenados
74	9	#####	73	DLD	DLD		2.0		falta copiar una variable del formato template a el diagra uml
75	9	#####	74	DLD	DLD		2.0		Faltaba iniciar la variable bool negative de la funcion reglasmpror en false
76	9	#####	75	DLD	CODE		1.0		en el template de diseño, la funcion REGLASMPX2 no tenia el tipo double de la funcion
77	9	#####	76	CODE	CR		1.0		punto y coma en la linea 345
78	9	#####	77	CODE	CR		2.0		Cambio de variable tabla valoresnor como global
79	9	#####	78	CODE	CR		1.0		punto y coma en la linea 375
80	9	#####	79	CODE	COMPILE		2.0		cambiar el dato de datox a numerox de la estructura en la funcion de ordenamiento.
81	9	#####	80	CODE	TEST		28.0		en la linea 317 al hacer la division era por 2.0 por el tipo de dato.
82									
83									
84									

### source program listing

/\*Version: 1.2

Nombre: Juan Alberto Gutierrez Canto

Fecha: 14/07/2016

Descripcion: Programa 9A de la materia proceso personal de software,  
programa que calcula  
normalizacion de datos de  $x^2$

+++++  
\*/

/\*+++++Contenido+++++

LOC reusadas: 35

LOC modificadas: 15

LOC compilación : 566

Librerias:

#include <iostream>

#include <stdio.h>

#include <cstdlib>

#include <malloc.h>



```
#include <math.h>
```

```
#define PI 3.1416
```

Clases: N/A

Funciones:

PEDIRDATOS()

CALCULARRANGO()

XI()

XI2()

EXPX()

FXI()

TERM()

RESULTADOS()

Surce code in C:\Users\equipo\Documents\Personal Process

Software\5A\main.cpp

```
+++++  
*/
```

```
#include <iostream>
```

```
#include <stdio.h>
```

```
#include <cstdlib>
```

```
#include <malloc.h>
```

```
#include <math.h>
```

```
#define PI 3.1416
```

```
using namespace std;
```

```
double tablavaloresnor[10][10];
```

```
/*+++++ REUTILIZACION
```

```
+++++
```

```
Function: struct nodo()
```

Propósito: estructura para guardar dos datos doubles, con una liga

In/out: N/A

Limitaciones: ninguna

```
+++++  
+++++  
*/
```

```
struct nodo{
```

```
    double numerox;
```

```
    double numeroy;
```

```
    struct nodo *sig;
```

```
}; struct nodo *raiz=NULL, *last=NULL;
```

```
/*+++++reutilizado +++++
```



**Function: ISEMPTY()****Propósito:** revisar si la lista esta vacia**In/out:** no aplica/1 esta vacia-0 no esta vacia**Limitaciones:** no aplica

```
+++++
*/
```

```
int ISEMPTY()
{
    if(raiz== NULL)
        return 1;
    return 0;
}
```

```
/*+++++reutilizado+++++
```

```
void INSERT(double dato)
```

**Descripcion:** insertar un dato en un nuevo nodo**Limitaciones:** tipo de dato double**Input:** double datox,datoy;**Output:** no aplica;

```
+++++
*/
```

```
void INSERT(double datox)
{
    struct nodo *nuevo=NULL;
    if(ISEMPTY())
    {
        nuevo=(struct nodo *)malloc(sizeof(struct nodo));
        nuevo->numerox=datox;
        nuevo->sig=NULL;
        raiz=last=nuevo;
    }
    else
    {
        nuevo=(struct nodo *)malloc(sizeof(struct nodo));
        nuevo->numerox=datox;
        nuevo->sig=NULL;
        last->sig=nuevo;
        last=nuevo;
    }
}
```

```
/*+++++++reutilizado+++++++
```

```
double REMUEVE()
```

```
Descripcion: emueve datos de la lista elimina el nodo
```

```
Limitaciones: tipo de dato double
```

```
Input: no aplica
```

```
Output: datos
```

```
+++++
```

```
*/
```

```
void REMUEVE()
```

```
{
```

```
    struct nodo *elimina=NULL;
```

```
    if(!ISEMPTY())
```

```
    {
```

```
        elimina=raiz;
```

```
        raiz=raiz->sig;
```

```
        free(elimina);
```

```
    }
```

```
}
```

```
/*+++++++ seccion
```

```
+++++
```

```
Function: PEDIRDATOS()
```

```
Propósito: pedir los datos que se van a utilizar para las formulas y asignar  
datos duros
```

```
In/out: N/A
```

```
Limitaciones: ninguna
```

```
+++++
```

```
+++++
```

```
*/
```

```
int PEDIRDATOS()
```

```
{
```

```
    double datox;
```

```
    double datoy;
```

```
    long long n;
```

```
    long long i;
```

```
    printf("Introduce el numero de datos n\n");
```

```
    cin>>n;
```

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

```
    {
```

```
        printf("dame el numero %d\n",i+1);
```

```
        cin>>datox;
```

```
    INSERT(datox);  
}  
return n;  
}
```

```
/*+++++++ REUTILIZACION
```

```
+++++++
```

```
Function: BURBUJALTST()
```

```
Propósito: ordena los datos de una lista ligada
```

```
In/out: n de datos
```

```
Limitaciones: ninguna
```

```
+++++++
```

```
+++++++
```

```
*/
```

```
void BURBUJALTST(int n)
```

```
{  
    struct nodo *inicio;  
    struct nodo *auxiliar;  
    double dato;  
    long i;  
    long y;  
    double datoy;  
    i=0;  
    y=0;  
    inicio=raiz;  
    while(i<n)  
    {  
        y=i+1;  
        auxiliar=inicio->sig;  
        while(y<n)  
        {  
            if(true)  
            {  
                if(inicio->numero>auxiliar->numero)  
                {  
                    dato=auxiliar->numero;  
                    datoy=auxiliar->numery;  
                    auxiliar->numero=inicio->numero;  
                    auxiliar->numery=inicio->numery;  
                    inicio->numero=dato;  
                    inicio->numery=datoy;  
                }  
            }  
        }  
    }  
}
```

```
    }
    else
    {
        if(inicio->numeroy>auxiliar->numeroy)
        {
            datox=auxiliar->numerox;
            datoy=auxiliar->numeroy;
            auxiliar->numerox=inicio->numerox;
            auxiliar->numeroy=inicio->numeroy;
            inicio->numerox=datox;
            inicio->numeroy=datoy;
        }
    }
    y++;
    auxiliar=auxiliar->sig;
}
i++;
inicio=inicio->sig;

}
}

/*+++++++ REUTILIZACION
+++++++
Function: AVG()
Propósito: calcula el average de una serie de numeros
In/out: n numeros/average
Limitaciones: ninguna

+++++++
+++++++
*/

double AVG(int n)
{
    struct nodo *inicio=NULL;
    double avgdata;
    long i;
    inicio=raiz;
    avgdata=0;
    for(i=0;i<n;i++)
    {
        avgdata+=inicio->numerox;
        inicio=inicio->sig;
```

```
}
    avgdata=avgdata/n;
    return avgdata;
}
```

```
/*+++++++ REUTILIZACION
```

```
+++++
```

Function: **VARIANZA()**

Propósito: calcula la varianza de n datos

In/out: n datos, average /varianza

Limitaciones: ninguna

```
+++++
```

```
+++++
```

```
*/
```

```
double VARIANZA(int n,double avgdata)
```

```
{
    struct nodo *inicio=NULL;
    double varianza;
    long i;
    inicio=raiz;
    varianza=0;
    for(i=0;i<n;i++)
    {
        varianza+=pow((inicio->numeros - avgdata),2);
        inicio=inicio->sig;
    }
    varianza=varianza/(n-1);
    return varianza;
}
```

```
/*+++++++ REUTILIZACION
```

```
+++++
```

Function: **CALRANGOS()**

Propósito: calcula los segmentos que se pueden usar para la  
normalizacion

In/out: n numero de datos/numero de segmentos

Limitaciones: ninguna

```
+++++
```

```
+++++
```

```
*/
```

```
int CALRANGOS(int n)
{
    int s;
    int i;
    if(n%5==0)
    {
        for(i=0;i<(n/5);i++)
        {
            tablavaloresnor[3][i]=5;
        }
        return n/5;
    }
    else if(n%5==4)
    {
        for(i=0;i<(n/5);i++)
        {
            tablavaloresnor[3][i]=5;
        }
        tablavaloresnor[3][i]=4;
        return (n/5)+1;
    }
    else if(n%5<4)
    {
        for(i=0;i<((n/5)-1);i++)
        {
            tablavaloresnor[3][i]=5;
        }
        for(i=0;i<((n%5)+5);i++)
        {
            tablavaloresnor[3][i+1]++;
        }
        return (n/5)-1;
    }
}
```

/\*+++++++ REUTILIZAR

+++++++

Function: REGLASMPNOR()

Propósito: calcula la regla de simpson de la forma normal

In/out: valor de xlow, xhigh, numero de bloques/resultados

Limitaciones: ninguna



+++++

+++++

\*/

double REGLASMPNOR(double xlow,double xhigh,double N)

```
{
    double w;
    double fija;
    double x2;
    double ex;
    double fx;
    double term;
    double sumterm;
    bool negative;
    int i;
    sumterm=0;
    negative=false;
    fija=1.0/sqrt(2*PI);
    if(xhigh<0)
    {
        xhigh=xhigh*(-1);
        negative=true;
    }
    w=xhigh/N;
    for(i=0;i<=20;i++)
    {
        x2=pow(xlow,2)/2.0;
        ex=exp(-x2);
        fx=ex*fija;
        if(i==0||i==20)
        {
            term=fx*(w/3.0);
        }
        else if(i%2==0)
        {
            term=2.0*fx*(w/3.0);
        }
        else
        {
            term=4.0*fx*(w/3.0);
        }
        sumterm+=term;
        xlow+=w;
    }
}
```

```
if(negative)
{
    sumterm-=.5;
    sumterm*=-1;
}
else
{
    sumterm+=.5;
}
return sumterm;
}
```

```
/*+++++++ REUTILIZABLE
```

```
+++++++
```

```
Function: SECNORMAL()
```

```
Propósito: calcula las secciones de la formula para la normalizacion.
```

```
In/out: no de datos/calidad q
```

```
Limitaciones: ninguna
```

```
+++++++
```

```
+++++++
```

```
*/
```

```
double SECNORMAL(int n)
{
    struct nodo *inicio=NULL;
    double sumterm;
    double rglsmpls;
    long segmentos;
    double pk;
    double contsegment;
    double pabove;
    double pbelow;
    double xbelow;
    double xabove;
    double busqueda;
    double contador;
    int i;
    segmentos=CALRANGOS(n);
    rglsmpls=-4;
    sumterm=0;
    contsegment=0;
    pabove=0;
    inicio=raiz;
```



```
busqueda=inicio->numeroy;
tablavaloresnor[1][0]=-100000;
for(i=0;i<segmentos;i++)
{
    contsegment+=tablavaloresnor[3][i];
    pk=(1.0/n)*contsegment;
    do
    {
        pbelow=pabove;
        xbelow=xabove;
        pabove=REGLASMPNOR(0,rglsmps,20);
        xabove=rglsmps;
        rglsmps+=.1;
    }while(pabove<pk);
    tablavaloresnor[2][i]=xbelow+((pk-pbelow)/(pabove-pbelow))*(xabove-
xbelow);
    tablavaloresnor[1][i+1]=tablavaloresnor[2][i];
    contador=0;
    while(busqueda<tablavaloresnor[2][i])
    {
        contador++;
        if(inicio->sig==NULL)
            break;
        inicio=inicio->sig;
        busqueda=inicio->numeroy;
    }
    tablavaloresnor[4][i]=contador;
    tablavaloresnor[5][i]=pow(tablavaloresnor[3][i]-tablavaloresnor[4][i],2);
    tablavaloresnor[6][i]=tablavaloresnor[5][i]/tablavaloresnor[3][i];
    sumterm+=tablavaloresnor[6][i];
}
return sumterm;
}
```

/\*+++++++ REUTILIZABLE

+++++++

Function: REGLASPMX2()

Propósito: calcula la regla de simpson para la forma  $x^2$

In/out: valor de xlow, xhigh, numero de bloques, grados de libertad/

resultado de la grafica

Limitaciones: ninguna



+++++

+++++

\*/

double REGLASPMX2(double xlow,double xhigh,double N,double n)

```
{
    double w;
    double tfija;
    double fija;
    double xan;
    double exa;
    double fx;
    double term;
    bool negative;
    int i;
    double sumterm;
    negative=false;
    tfija=1;
    sumterm=0;
    for(i=1;i<n;i+=2)
    {
        tfija=tfija*(i/2.0);
    }
    tfija=tfija*sqrt(PI);
    fija=pow(2,(n/2));
    if(xhigh<0)
    {
        xhigh=xhigh*-1;
        negative=false;
    }
    w=xhigh/N;
    for(i=0;i<=N;i++)
    {
        xan=pow(xlow,(n/2)-1);
        exa=exp(-(xlow/2));
        fx=(xan*exa)/(fija*tfija);
        if(i==0||i==20)
        {
            term=fx*(w/3);
        }
        else if(i%2==0)
        {
            term=2*fx*(w/3);
        }
    }
}
```

```
    else
    {
        term=4*fx*(w/3);
    }
    sumterm+=term;
    xlow+=w;
}
if(negative)
{
    sumterm-=.5;
}
else
{
    sumterm+=.5;
}
return sumterm;
}
```

```
/*+++++++ REUTILIZABLE
```

```
+++++++
```

Function: **PROBABILIDAD()**

Propósito: calcula la probabilidad 1-p

In/out: prob numero a calcular

Limitaciones: ninguna

```
+++++++
```

```
+++++++
```

```
*/
```

```
double PROBABILIDAD(double prob)
```

```
{
    prob=1-prob;
    return prob;
}
```

```
/*+++++++ REUTILIZABLE
```

```
+++++++
```

Function: **NORMALIZACION()**

Propósito: calcula la calidad de los datos normalizando

In/out: n numero de datos/ probabilidad

Limitaciones: ninguna

```
+++++++
```

```
+++++++
```

\*/

```
double NORMALIZACION(int n)
{
    double xavg;
    double varianza;
    double dest;
    double q;
    double simx2;
    double p;
    struct nodo *inicio=NULL;
    int i;
    double aux;
    inicio=raiz;
    xavg=AVG(n);
    varianza=VARIANZA(n,xavg);
    varianza=sqrt(varianza);
    for(i=0;i<n-1;i++)
    {
        aux=(inicio->numerox -xavg)/varianza;
        inicio->numeroy=aux;
        inicio=inicio->sig;
    }
    q=SECNORMAL(n);
    cout<<"q "<<q<<endl;
    p=REGLASPMX2(0,q,20,9);
    p=p-REGLASPMX2(0,q,40,9);
    p=1-p;
    p=PROBABILIDAD(p);
    return p;
}
```

/\*+++++++ seccion

+++++++

Function: main()

Propósito: funcion principal

In/out: N/A

Limitaciones: ninguna

+++++++

+++++++

\*/

int main()

```
{  
    int n;  
    freopen("prueba2.txt","r",stdin);  
    n=PEDIRDATOS();  
    BURBUJALTST(n);  
    cout<<"resultado "<<NORMALIZACION(n);  
    return 0;  
}
```

### Test result

Test	Expectec result	Actual result
Table d14		
Q	34.4	34.4
1-p	$7.60 \cdot 10^{-5}$	0.00301576

### Diseño.

