



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 7A
Fecha	14 de Junio 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.**

Project

7

Import

### PSP2 Project Plan Summary - Program 7A

Student 94419-JUAN ALBERTO GUTIER Date \*\*\*\*\*  
 Program Correlation and Significance Program# 7A  
 Instructor PATRICIA ERENDIRA BENAVIDES Language C

	Plan	Actual	To Date	To Date%
<b>Summary</b>				
LOC/Hour	28.62	23.92	23.92	
Planned Time	271		998	
Actual Time		534	1369	
CPI (Planned/Actual Time)			0.73	
% Reused	33.7	29.6	29.6	
% New Reused	0.0	0.0	0.0	
<b>Test Defects/Ki</b>	<b>10.38</b>	<b>9.3897</b>	<b>9.9897</b>	
<b>Total Defects/Ki</b>	<b>13.346</b>	<b>98.592</b>	<b>98.592</b>	
<b>Yield %</b>	<b>89.744</b>	<b>90.476</b>	<b>90.476</b>	

### Program Size (LOC)

Base(B)	10	16	
Deleted(D)	1	5	
Modified(M)	30	50	
Added(A)	99	163	
Reused(R)	55	73	73
Total N&C (N)	129	213	213
Total LOC(T)	163	247	247
Total New Reused	0	0	0
Total Object LOC(E)	165.43	197	197
<b>UPI (70%)</b>	<b>169</b>		
<b>LPI (70%)</b>	<b>89</b>		

### Time in Phase (min.)

Planning	37	93	93	17.5
Design	27	216	216	40.5
<b>Design Review</b>	<b>39</b>	<b>37</b>	<b>37</b>	<b>7.0</b>
Code	50	95	95	17.8
<b>Code Review</b>	<b>39</b>	<b>25</b>	<b>25</b>	<b>4.7</b>
Compile	0	1	1	0.2
Test	21	36	36	6.7
Postmortem	43	30	30	5.6
Total	271	534	534	100.0
<b>UPI (70%)</b>	<b>322</b>			
<b>LPI (70%)</b>	<b>220</b>			

### PSP2 Project Plan Summary - Program 7A (Continued)

Student 94419-JUAN ALBERTO GUTIER Date #####  
 Program Prediction Interval Program# 7A  
 Instructor PATRICIA ERENDIRA BENAVIDES Language C

	Plan	Actual	To Date	To Date%
<b>Defects Injected</b>				
Planning	0.2272	0	0	0.0
Design	1.1549	15	15	71.4
<i>Design Review</i>	0	0	0	0.0
Code	6.0014	6	6	28.6
<i>Code Review</i>	0	0	0	0.0
Compile	1.1549	0	0	0.0
Test	0.9277	0	0	0.0
Total Development	9.466	21	21	100.0

<b>Defects Removed</b>				
Planning	0	0	0	0.0
Design	0	0	0	0.0
<i>Design Review</i>	1.382	11	11	52.4
Code	0	4	4	19.0
<i>Code Review</i>	5.2442	4	4	19.0
Compile	1.5079	0	0	0.0
Test	1.3319	2	2	9.5
Total Development	9.466	21	21	100.0
After Development		21	21	

<b>Defect Removal Efficiency</b>			
<i>Def/Hr - DLDR</i>	0	0	0
<i>Def/Hr - CDR</i>	0	2.5201	2.5201
<i>Def/Hr - Comp</i>	12.02	0	0
<i>Def/Hr - Test</i>	3.7985	3.338	3.338
<i>DRL(DLDR/UT)</i>	1.0377	5.5	5.5
<i>DRL(CDR/UT)</i>	3.9375	2	2
<i>DRL(Compile)</i>	1.1322	0	0



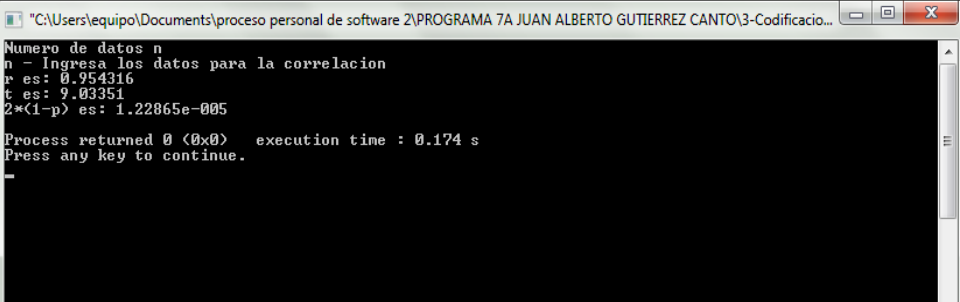
<b>Test Name/#</b>	1
<b>Test Objective</b>	Valores correlacion, significancia y probabilidad
<b>Test Description</b>	Encontrar los valores de correlacion, significación y probabilidad para un conjunto de 10 números x and y.
<b>Test Conditions</b>	10 186 15.0 699 69.9 132 6.5 272 22.4 291 28.4 331 65.9 199 19.4 1890 198.7 788 38.8 1601 138.2
<b>Expected Results</b>	R 0.9543 T 9.0335 $2*(1-p) 1.80*10^{-5}$
<b>Actual Results</b>	

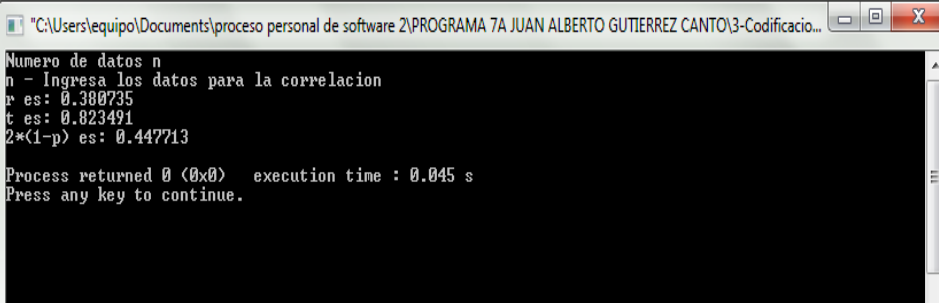
<b>Test Name/#</b>	2
<b>Test Objective</b>	Valores correlacion, significancia y probabilidad
<b>Test Description</b>	Encontrar los valores de correlacion, significación y probabilidad para un conjunto de 10 números x and y.
<b>Test Conditions</b>	10 186 15.0 699 69.9 132 6.5 272 22.4 291 28.4 331 65.9



## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

	199 19.4 1890 198.7 788 38.8 1601 138.2
<b>Expected Results</b>	R 0.9543 T 9.0335 $2*(1-p) 1.80*10^{-5}$
<b>Actual Results</b>	

<b>Test Name/#</b>	3
<b>Test Objective</b>	Valores correlacion, significancia y probabilidad
<b>Test Description</b>	Encontrar los valores de correlacion, significación y probabilidad de los valores Actual LOC versus Development Time.
<b>Test Conditions</b>	6 79 98.8 105 137.2 133 106.5 133 207.1 90 161.7 106 124.2
<b>Expected Results</b>	N/A
<b>Actual Results</b>	



--	--

Test Name/#	4
Test Objective	Valores correlacion, significancia y probabilidad
Test Description	Encontrar los valores de correlacion, significación y probabilidad de los valores Estimated LOC versus Development Time.
Test Conditions	6 0 98.8 250 137.2 250 106.554 207.1 32 161.7 54 124.2
Expected Results	N/A
Actual Results	

**Los documentos de:**

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

## PSP2 Design Review checklist

### PSP2 Design Review Checklist

Student	Juan Alberto Gutierrez Canto	Date	13/06/2016
Program	Correlacion lineal	Program #	7A
Instructor	Jorge Rafael Aguilar Cisneros	Language	C++

Purpose	To guide you in conducting an effective design review
---------	---



# PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

<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	B  X	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> <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  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		





## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

### Code Review checklist

#### Code Review Checklist

Student	Juan Alberto Gutierrez Canto	Date	13/06/2016
Program	Correlacion lineal	Program #	7A
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. <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



## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

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

### SIZE ESTIMATING TEMPLATE

Project 7	<b>Size Estimating Template</b>					
	Student Instructor	JUAN ALBERTO GUTIERREZ CAN Date JORGE RAFAEL AGUILAR CISNEF Program#			10/06/2016 7A	
<b>BASE PROGRAM LOC</b>				<b>ESTIMATE</b>	<b>ACTUAL</b>	
BASE SIZE (B) =>				10	16	
LOC DELETED (D) =>				1	5	
LOC MODIFIED (M) =>				30	50	
<b>OBJECT LOC</b>						
<b>BASE ADDITIONS:</b>				<b>LOC</b>	<b>LOC</b>	
includes & define				8.8	7	
variables globales				8.8	3	
(BA) subtotal from page 2				0.0	0	
TOTAL BASE ADDITIONS (BA)				17.7	10	
<b>NEW OBJECTS:</b>				<b>LOC *</b>	<b>LOC *</b>	
CALCULACORRELACION()				11.3	22	
CALCULART1				11.3	9	
CALCULART2				11.3	9	
CALCULART3				11.3	5	
REGLASMP				11.3	30	
ISEMPY()				11.3	6	
INSERT()				11.3	20	
REMUEVE()				11.3	9	
(NO) subtotal from page 2				27.6	27	
TOTAL NEW OBJECTS (NO)				117.8	137	
<b>REUSED OBJECTS</b>				<b>LOC</b>	<b>LOC</b>	
PEDIRDATOS()				6.0	16	
CALCULARRANGO()				6.0	9	
XII()				9.0	11	
XII2()				4.0	5	
EXPX()				4.0	5	
FXI()				4.0	5	
TERM(int i)				13.0	16	
(R) subtotal from page 2				9.0	6	
REUSED TOTAL (R)				55.0	73	

		Size	Time
Estimated Object LOC:	$E = BA + NO + M$	165.43	
Regression Parameter:	$B_0$	63.45	68.65
Regression Parameter:	$B_1$	0.40	0.82
Estimated New and Changed LOC:	$N = B_0 + B_1 * E$	129.1	
Estimated Total LOC:	$T = N + B - D - M + R$	163.1	
Estimated Total New Reuse (sum of * LOC):		0	
Estimated Total Development Time:	$Time = B_0 + B_1 * E$		204.5
Prediction Range:	Range	40.3	51.3
Upper Prediction Interval:	$UPI = N + Range$	169.4	255.8
Lower Prediction Interval:	$LPI = N - Range$	88.7	153.1
Prediction Interval Percent		70%	70%
Method Selected		A	A
R <sup>2</sup>		0.76	0.89

### Size Estimating Template (continued)

Student	JUAN ALBERTO GUTIERREZ CAN	Date	10/06/2016
Instructor	JORGE RAFAEL AGUILAR CISNEF	Program#	7A

[illegible]

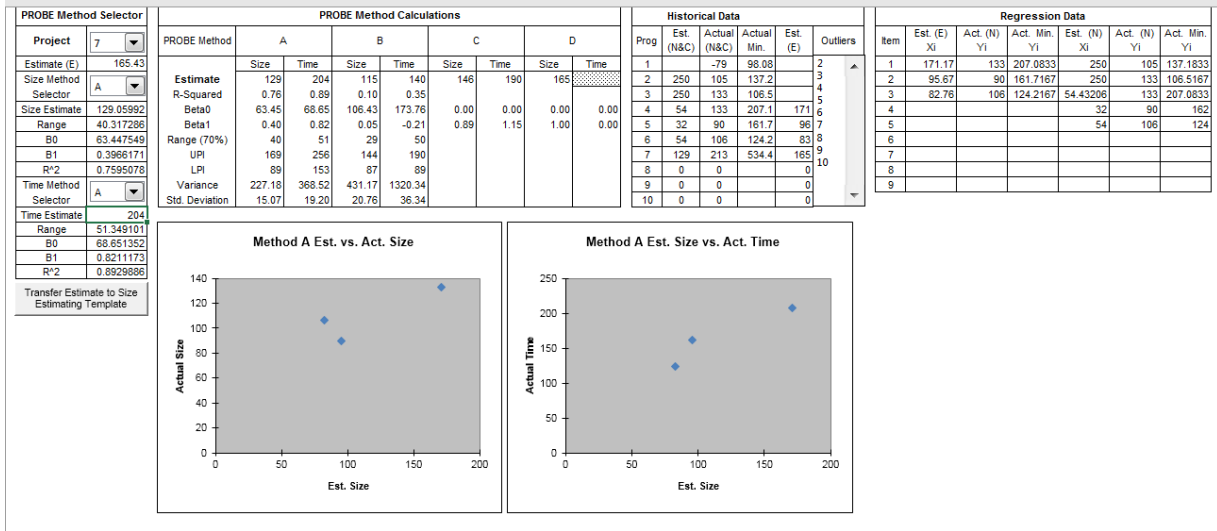
[illegible]

## PROBE Calculation worksheet



## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016



## Time Recording Log.

1	Project	Phase	Date	Start	Int.	Stop	Delta	Comments
43	5	TEST	04/04/16	12:51:57		13:16:40	24.7	CORRECCION DE ERRORES
44	5	PM	04/04/16	13:35:10		13:53:32	18.4	FINALIZADO EL PROGRAMA CON TODA LA DOCUMNTACION
45	5	PM	04/07/16	10:25:52		10:29:35	3.7	CORRECCION DE ERRORES
46	6	PLAN	04/07/16	10:30:24		11:00:12	29.8	TIEMPO DE IR A COMER
47	6	PLAN	04/07/16	13:14:02		13:24:33	10.5	COMPLETADO PLANACION
48	6	DLD	04/12/16	10:48:22		11:05:13	16.9	DISEÑO FINALIZADO
49	6	CODE	04/12/16	11:58:18		12:28:22	30.1	CODIGO TERMINADO
50	6	COMPILE	04/13/16	01:01:24		01:08:42	7.3	COMPILADO y correguido
51	6	TEST	04/13/16	01:09:24		01:20:04	10.7	PROBADO
52	6	PM	04/13/16	05:39:13		05:58:14	19.0	FINALIZADO EL PROGRAMA CON TODA LA DOCUMNTACION
53	7	PLAN	06/08/16	15:20:07		15:39:21	19.2	PLANEACION EN EL SALON DE CLASES
54	7	PLAN	06/09/16	15:00:46		15:39:54	39.1	PLANEACION EN EL SALON DE CLASES Y ENTENDIMIENTO DEL PROGRAMA
55	7	PLAN	06/10/16	00:27:30		01:02:33	35.1	TERMINO DE PLANEACION
56	7	DLD	06/10/16	01:04:49		01:25:28	20.7	COMIENZO DEL DISEÑO DIAGRAMA DE ACTIVIDADES
57	7	DLD	06/11/16	20:25:38		21:34:02	68.4	DIAGRAMA DE ESTADOS
58	7	DLD	06/12/16	13:37:06		14:58:36	81.5	DIAGRAMA DE CLASES
59	7	DLD	06/12/16	23:30:23		00:16:09	45.8	TERMINO DE DISEÑO
60	7	DLD	06/13/16	14:18:15		14:55:39	37.4	REVISION DE DISEÑO
61	7	CODE	06/13/16	15:04:09		15:58:58	54.8	CODIFICACION CON LOS DISEÑOS
62	7	CODE	06/13/16	23:04:49		23:45:14	40.4	COMPLEMENTO DE ESTANDAR DE CODIFICACION Y TERMINO DE CODIFICACION
63	7	CR	06/14/16	00:51:48		01:17:02	25.2	REVISION DE CODIGO LLENANDO EL TEMPLATE
64	7	COMPILE	06/14/16	01:21:19		01:22:22	1.1	COMPILACION SIN ERRORES
65	7	TEST	06/14/16	01:31:50		02:07:47	36.0	TERMINO DE PRUEBAS Y CORRECCION DE ERRORES
66	7	PM	06/14/16	02:08:24		02:38:11	29.8	LLENADO DE TEMPLATES
67								
68								

## Defect Recording Log.





## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

#	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Project	Date	Num	Type	Injected	Removed	FixTime	Fix Ref.	Description							
44	5	#####	43	10	COMPILE	PM	1.0		El defecto 37 es de interface							
45	6	#####	44	20	CODE	COMPILE	1.0		Me faltaba un * en la linea 170 para hacer la multiplicacion de las variables							
46	6	#####	45	20	CODE	COMPILE	1.0		Me faltaba un * en la linea 179 para hacer la multiplicacion de las variables							
47	6	#####	46	80	DLD	COMPILE	5.0		Para calcular upi y lpi se tenia que aplicar la formula $y_k=b_0+b_1x_k$							
48	6	#####	47	80	DLD	TEST	2.0		En el caso de prueba me faltaba introducir un valor inicial al documento txt							
49	7	#####	48	80	DLD	DLDL	0.5		me faltaba agregar la regla de que cuando la funcion de simpson no alcanza el error los bloques se incrementan al doble							
50	7	#####	49	40	DLD	DLDL	0.5		inicializar en el diseño el numero de bloques N como 20							
51	7	#####	50	40	DLD	DLDL	1.0		inicializacion de error en 0.00001							
52	7	#####	51	40	DLD	DLDL	1.0		inicializacion de xlow en 0							
53	7	#####	52	40	DLD	DLDL	1.0		inicializacion de pi en 3.1416							
54	7	#####	53	10	DLD	DLDL	1.0		cambio de nombre de la variable de anterior a pasado en el diagrama de estados							
55	7	#####	54	10	DLD	DLDL	1.0		cambio de nombre de la variable de N a Nbloques en el diagrama de estados							
56	7	#####	55	40	DLD	DLDL	1.0		declarar variables x y y dentro del diagrama de clases							
57	7	#####	56	80	DLD	DLDL	2.0		faltaba sacar la raiz de la correlacion							
58	7	#####	57	80	DLD	DLDL	1.0		sobraba el metodo sqrt() en la clase EXPX() de el diagram de clases							
59	7	#####	58	80	DLD	DLDL	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	10	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	10	DLD	CODE	1.0		la variable l de el diagrama de clases en la funcion significancia se me olvido borrarla							
62	7	#####	61	10	DLD	CODE	1.0		la funcion CALCULART3 le sobraba una R en el diagrama de clases							
63	7	#####	62	10	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	80	CODE	CR	3.0		en la linea 187 me faltaba un parentesis en la formula							
65	7	#####	64	80	CODE	CR	1.0		en la funcion correlacion habia un return 0 de mas							
66	7	#####	65	80	CODE	CR	1.0		faltaba un par de parentesis en la funcionEXPX en la formula del valor							
67	7	#####	66	80	CODE	CR	1.0		faltaba agregar la funcion de archivo freopen en el main							
68	7	#####	67	80	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	80	CODE	TEST	15.0		falba inicializar la regla de simpson por cada iteracion							
70																
71																

### source program listing

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

**Version: 1.0**

**Nombre: Juan Alberto Gutierrez Canto**

**Fecha: 13/06/2016**

**Descripcion: Programa 7A de la materia proceso personal de software, programa que calcula**

**la correlacion lineal de 2 series de numeros y su significancia y probabilidad**

```
+++++
```

```
*/
```

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

**LOC reusadas: 105**

**LOC modificadas: 50**

**LOC compilación : 229**

**Librerias:**

```
#include <cstdlib>
```

```
#include <iostream>
```

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

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

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

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

```
#define PI 3.1416
```

**Clases: N/A**

**Funciones:**

```
int ISEMPTY()
```

```
void INSERT(double datox,double datoy)
```

```
void REMUEVE()
```

```
void PEDIRDATOS()
```

```
float CALCULARCORRELACION()
float SIGNIFICANCIA()
void CALCULART1()
void CALCULART2()
void CALCULART3()
void CALCULARRANGO()
float XI()
float XI2()
float EXPX()
float FXI()
float TERM(int i)
float REGLASMP()
float PROBABILIDAD()
void RESULTADO()
int main()
source: C:\Users\equipo\Documents\proceso personal de software
2\PROGRAMA 7A JUAN ALBERTO GUTIERREZ CANTO\3-Codificacion
+++++
*/
#include <iostream>
#include<stdlib.h>
#include <malloc.h>
#include <math.h>
#include <stdio.h>
#include <cstdlib>
#define PI 3.1416

using namespace std;
double xlow,xhigh,w,pasado,anterior,error,rt1,rt2,rt3,t,r,p;
int Nbloques,Nitems,n;
bool primera=false;

struct nodo
{
    /*-c estructura para guardar las listas ligadas*/
    double numerox,numeroy;
    struct nodo *sig;
};struct nodo *raiz=NULL, *last=NULL;

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



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

```
int ISEMPY()
{
    /*-c funcion para saber si esta vacia las listas ligadas*/
    if(raiz == NULL)          /*-c esta vacia la lista?? 1-si 0-no*/
        return 1;
    else
        return 0;
}
```

```
/*+++++seccion+++++
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, double datoy)
{
    /*-c insertar un dato en un nuevo nodo*/
    struct nodo *nuevo = NULL;          /*-c nuevo nodo*/
    if(ISEMPY())
    {
        nuevo = (struct nodo *)malloc( sizeof (struct nodo)); /*-c inserta si
no hay ninguna lista todavia*/
        nuevo->numerox = datox;
        nuevo->numeroy=datoy;
        nuevo->sig = NULL;
        raiz = last = nuevo;
    }
    else
    {
        nuevo = (struct nodo *)malloc( sizeof (struct nodo)); /*-c insertar
cuando ya existe un nodo*/
        nuevo->numerox = datox;
        nuevo->numeroy=datoy;
        nuevo->sig = NULL;
        last->sig = nuevo;
        last = nuevo;
    }
}
```

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

```
double REMUEVE()
```

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

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

```
Input: no aplica
```

```
Output: datos
```

```
+++++
```

```
*/
```

```
void REMUEVE()
```

```
{ /*-c regresar dato de la lista ligada*/
```

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

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

```
{
```

```
elimina = raiz;
```

```
raiz = raiz->sig; /*-c o raiz = elimina -> sig;*/
```

```
free(elimina); /*-c libera espacio de memoria*/
```

```
}
```

```
}
```

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

```
+++++
```

```
Function: PEDIRDATOS()
```

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

```
In/out: tasa distribucion
```

```
Limitaciones: ninguna
```

```
+++++
```

```
+++++
```

```
*/
```

```
void PEDIRDATOS()
```

```
{
```

```
double x,y;
```

```
int i;
```

```
xlow=0;
```

```
Nbloques=20;
```

```
error=0.000001;
```

```
cout<<"Numero de datos n"<<endl<<"n - ";
```

```
cin>>Nitems;
```

```
cout<<"Ingresa los datos para la correlacion\n";
```

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

```
{
```

```
        cin>>x>>y;
        INSERT(x,y);
    }
    pasado=0;
}

/*+++++++ seccion
+++++++
Function: CALCULARCORRELACION()
Propósito: calcula la correlacion de dos series de datos
In/out: series de datos x, y/ r correlacion lineal
Limitaciones: ninguna

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

double CALCULARCORRELACION()
{
    double x,y,val;
    double sumxcuadro=0,sumycuadro=0;
    double sumxy=0,sumx=0,sumy=0;
    int i,j;
    struct nodo *check=NULL;
    check=raiz;
    for(i=0;i<Nitems;i++)
    {
        x=check->numerox;
        y=check->numeroy;
        sumxy+=x*y;
        sumx+=x;
        sumy+=y;
        sumxcuadro+=x*x;
        sumycuadro+=y*y;
        check=check->sig;
        REMUEVE();
    }
    val=((n*sumxy)-(sumx*sumy))/(sqrt((((n*sumxcuadro)-
(pow(sumx,2)))*((n*sumycuadro)-(pow(sumy,2))))));
    return (val);
}

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



**Function: SIGNIFICANCIA()**

**Propósito: calcula significancia de los valores de correlacion**

**In/out: r correlacion lineal/signifiacnia t**

**Limitaciones: ninguna**

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

**double SIGNIFICANCIA()**

```
{
    double absr=r;
    if(r<0)
        absr=r*(-1);
    return ((absr*sqrt(n-2))/(sqrt(1-pow(r,2))));
}
```

**/\*+++++ seccion**

**+++++**

**Function: CALCULART1()**

**Propósito: calcula el valor gama 1 para la distribucion t de students**

**In/out: n grados de tolerancia / gama 1 rt1**

**Limitaciones: ninguna**

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

**void CALCULART1()**

```
{
    double val=1,a=2;
    int i;
    for(i=1;i<n;i+=2)
    {
        val*=i/a;
    }
    rt1=val*sqrt(PI);
}
```

**/\*+++++ seccion**

**+++++**

**Function: CALCULART2()**

**Propósito: calcula el valor gama 2 para la distribucion t de students**

**In/out: n grados de tolerancia / gama 2 rt2**

**Limitaciones: ninguna**

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

```
void CALCULART2()
{
    double val=1;
    int i;
    for(i=2;i<((n+1)/2);i++)
    {
        val*=i;
    }
    rt2=val;
}
```

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

```
+++++
```

**Function: CALCULART3()**

**Propósito: calcula el valor gama 3 para la distribucion t de students**

**In/out: n grados de tolerancia / gama 3 rt3**

**Limitaciones: ninguna**

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

```
void CALCULART3()
{
    double val;
    val=PI*n;
    rt3=sqrt(val);
}
```

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

```
+++++
```

**Function: CALCULARRANGO()**

**Propósito: calcula el rango y los bloques**

**In/out: no aplica/no aplica**

**Limitaciones: ninguna**

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

\*/

**void CALCULARRANGO()**

```
{
    double xhigh_abs;
    xhigh_abs=xhigh;
    if(xhigh<0)
        xhigh_abs=xhigh*(-1);
    w=xhigh_abs/Nbloques;
    pasado=0;
    primera=true;
}
```

/\*+++++++ Resuso

+++++++

**Function: XI()****Propósito: calcula los valores de la formula xi****In/out: no aplica/xi****Limitaciones: ninguna**

+++++++

+++++++

\*/

**double XI()**

```
{
    double xi=0;
    if(!primera)
    {
        primera=true;
        return 0;
    }
    xi=pasado+w;    //-c incrementamso w al rango
    pasado=xi;
    return xi;
}
```

/\*+++++++ seccion

+++++++

**Function: XI2()****Propósito: calcula los datos de la formula  $1+xi^2/n$** **In/out: no aplica/xi2****Limitaciones: ninguna**

+++++

\*/

**double XI2()**

{

**double xi2;**

**xi2=1+((pow(XI(),2))/n);**                   *//-c se eleva al cuadrado xi*

**return xi2;**

}

*/\*+++++ Reuso*

+++++

**Function: EXPX()**

**Propósito:** *calcula el valor de XI2() elevado a  $-(n+1/2)$*

**In/out:** *no aplica/valor*

**Limitaciones:** *ninguna*

+++++

+++++

*\*/*

**double EXPX()**

{

**double valor;**

**valor=pow(XI2(),(-(n+1)/2));**                   *//-c exponencial de xi*

**return valor;**

}

*/\*+++++ Resuso*

+++++

**Function:FXI()**

**Propósito:** *calcula fx para la distribucion t students*

**In/out:** *no aplica/fx*

**Limitaciones:** *ninguna*

+++++

+++++

*\*/*

**double FXI()**

{

**double fx;**

**fx= (rt2/(rt3\*rt1))\*EXPX();**                   *//-c calculamos la raiz de un numero y*

**sirve de divisro para un exponencial**

**return fx;**

```
}
```

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

```
+++++
```

```
Function: TERM()
```

```
Propósito: calcula el termino de la distribucion
```

```
In/out: int i/ float term
```

```
Limitaciones: ninguna
```

```
+++++
```

```
+++++
```

```
*/
```

```
double TERM(int i) //c recibe la posicion del dato
```

```
{
```

```
    double term,a=2,b=3,c=4;
```

```
    if(i== 0 || i== Nbloques) //c calculamos term con las reglas de que  
    si son la pocicion inicial y final se completa la siguiente formula
```

```
    {
```

```
        term=(FXI()*w)/b;
```

```
    }
```

```
    else if(i%2 == 0) //c los datos no son iniciales, por lo tanto  
    revisamos si son pares
```

```
    {
```

```
        term=a*FXI()*w/b;
```

```
    }
```

```
    else //c los datos no son pares
```

```
    {
```

```
        term=c*FXI()*w/b;
```

```
    }
```

```
    return term;
```

```
}
```

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

```
+++++
```

```
Function: REGLASMP()
```

```
Propósito: calcula el la regal de simpson para la distribucion t de students
```

```
In/out: no aplica/ float pasado
```

```
Limitaciones: ninguna
```

```
+++++
```

```
+++++
```

```
*/
```

```
double REGLASMP()
```

```
{
```



```
double suma_term=0,nuevoerror,a=0.5;
int i;
anterior=0;
Nbloques=10;
do
{
    Nbloques*=2;
    CALCULARRANGO();
    anterior=suma_term;
    suma_term=0;
    primera=false;
    for(i=0;i<=Nbloques;i++) //c calculamos la distribucion para los 20
datos
    {
        suma_term+=TERM(i); //c realizamos la sumatoria de todos los
terminos de la distribucion
    }
    if(xhigh<0)
    {
        suma_term=a-suma_term; //c calculamos el resultado
restando a 0.5 que es la distribucion normal, la suma de la suma de los 20
terminos
    }
    else
    {
        suma_term=a+suma_term; //c calculamos el resultado
sumando a 0.5 que es la distribucion normal, la suma de la suma de los 20
terminos
    }
    nuevoerror=suma_term-anterior;
    if(nuevoerror<0)
        nuevoerror=nuevoerror*(-1.0);
    }while(nuevoerror > error);
    return anterior;
}
```

/\*+++++++ seccion

+++++++

Function: **PROBABILIDAD()**

Propósito: **calcula la probabilidad de la distribucion t de students**

In/out: **no aplica/ float val**

Limitaciones: **ninguna**

+++++++

+++++++

\*/

**double PROBABILIDAD()**

{

**double val,a=2,b=1;****val=a\*(b-p);****return val;**

}

/\*+++++++ seccion

+++++++

**Function: RESULTADO()****Propósito: cimprime los resultados, correlacion r, significancia t,  
probabilidad p****In/out: no aplica/ no aplica****Limitaciones: ninguna**

+++++++

+++++++

\*/

**void RESULTADO()**

{

**cout<<"r es: "<<r<<endl;                //c imprimimos resultados****cout<<"t es: "<<t<<endl;                //c imprimimos resultados****cout<<"2\*(1-p) es: "<<p<<endl;                //c imprimimos resultados**

}

**int main()**

{

**freopen("prueba1.txt","r",stdin);****PEDIRDATOS();****n=Nitems;****r=CALCULARCORRELACION();****t=SIGNIFICANCIA();****n=Nitems-1;****CALCULART1();****CALCULART2();****CALCULART3();****xhigh=t;****p=REGLASMP();****p=PROBABILIDAD();****RESULTADO();****return 0;**

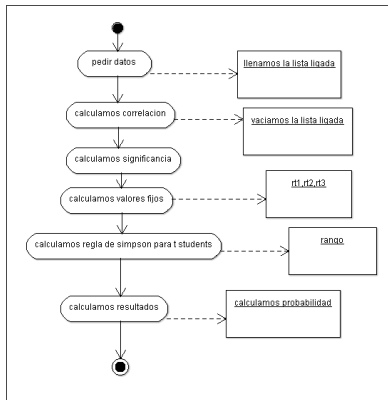
}

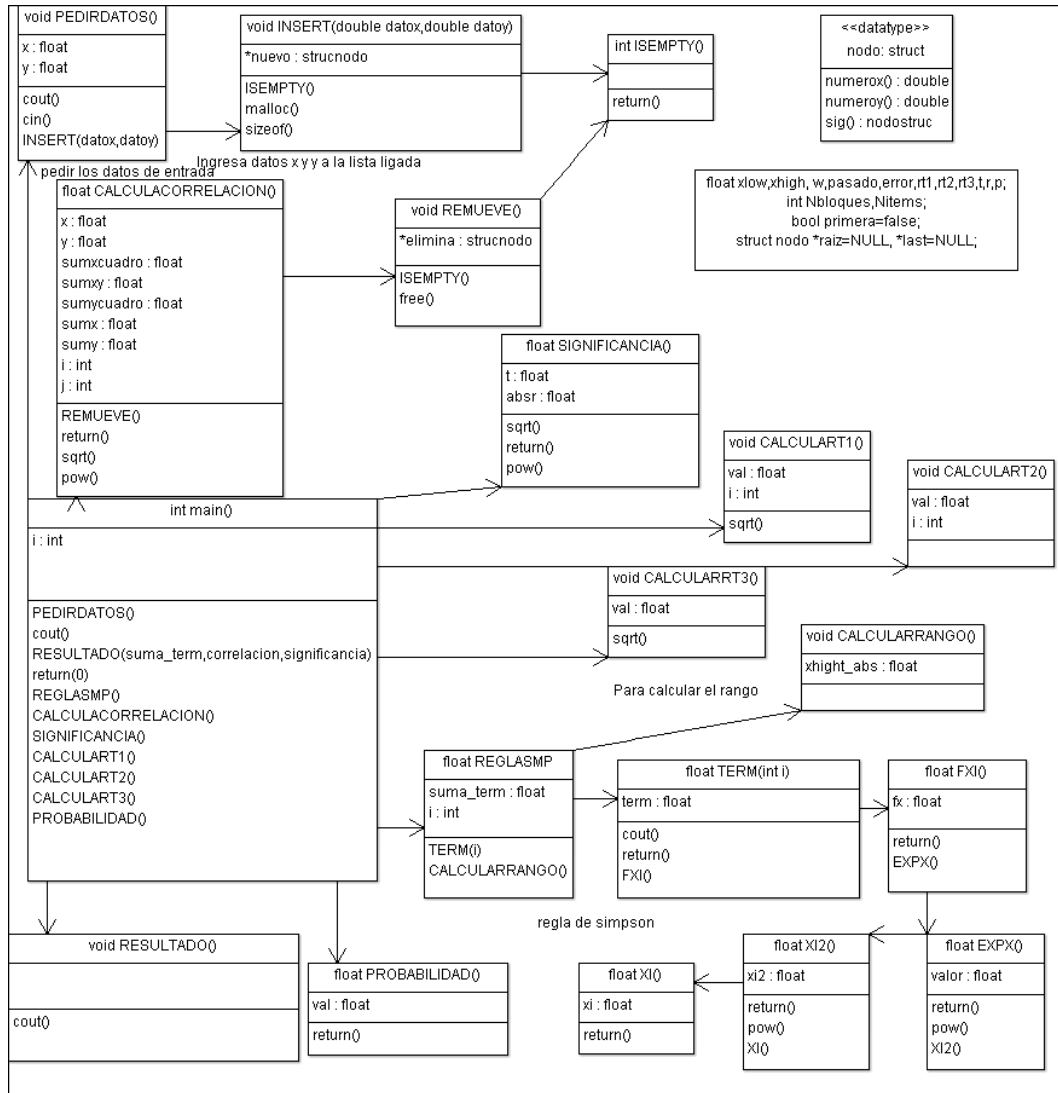


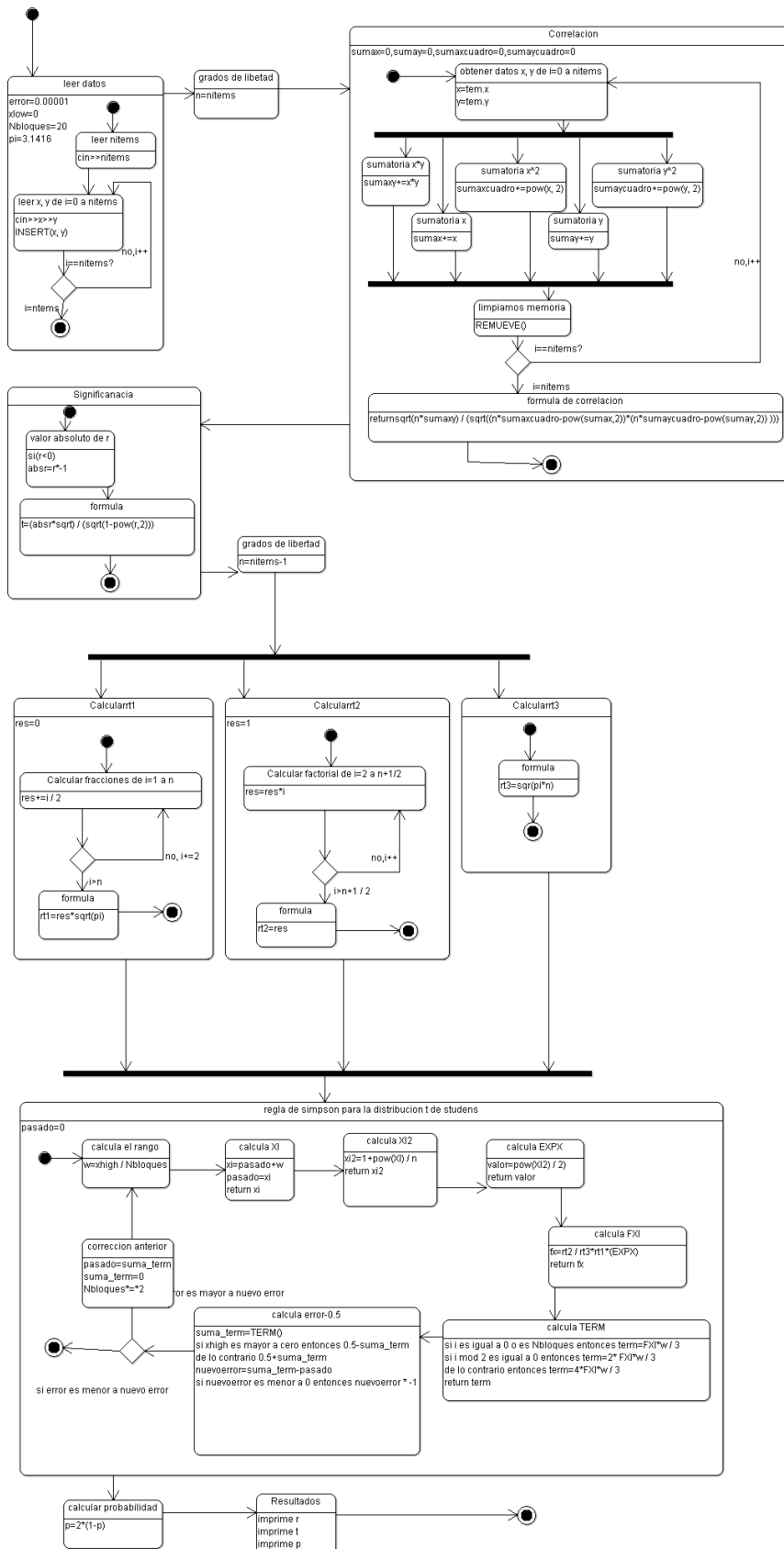
### Test result

Test	Expected Value			Actual Value		
	R	T	$2*(1-p)$	R	T	$2*(1-p)$
Table D12	0.9543	9.0335	$1.80*10^{-5}$	0.954316	9.03351	1.22865e-005
Actual LOC versus Development Time.	N/A	N/A	N/A	0.380735	0.823491	0.447713
Estimated LOC versus Development Time.	N/A	N/A	N/A	0.165507	0.335642	0.750767

### Diseño.









PROCESO PERSONAL DE DESARROLLO DE  
SOFTWARE 2

VERANO 2016