



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 8A
Fecha	26 de Junio de 2016

### Checklist de Correcciones.

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

## Forma PSP2 Project Plan Summary.

Import:

Student 34413-JUAN ALBERTO GUTIER Date 20160601  
 Program List Sort Program# 8A  
 Instructor JORGE RAFAEL AGUILAR CISI Language C

	Plan	Actual	To Date	To Date%
<b>Summary</b>				
LOC/Hour	46.04	21.00	23.10	
Planned Time	34		1093	
Actual Time		209	1578	
CPI (Planned/Actual Time)			0.69	
% Reused	36.8	33.8	31.1	
% New Reused	0.0	0.0	0.0	
Test Defects/KLOC	9.39	0.00	6.99	
Total Defects/KLOC	98.59	54.79	87.41	
Yield %	90.5	100.0	92.0	
<b>1 Appraisal CG</b>	<b>11.731</b>	<b>24.395</b>	<b>24.395</b>	
<b>1 Failure CG</b>	<b>1.9359</b>	<b>1.9312</b>	<b>2.9418</b>	
<b>CG A/F Ratio</b>	<b>1.6424</b>	<b>1.249</b>	<b>1.349</b>	

### Program Size (LOC)

Base(B)	20	30	
Deleted(D)	5	5	
Modified(M)	10	10	
Added(A)	62	63	
Reused(R)	45	45	118
Total N&C (N)	72	73	286
Total LOC(T)	122	123	380
Total New Reused	0	0	0
Total Object LOC(E)	57.11	87	284
UPI (70%)	162		
LPI (70%)	0		

### Time in Phase (min.)

Planning	16	21	114	15.4
Design	38	112	328	44.2
Design Review	7	25	63	8.5
Code	17	18	114	15.3
Code Review	4	17	42	5.7
Compile	0	0	1	0.1
Test	6	4	40	5.4
Postmortem	5	11	41	5.5
Total	94	209	743	100.0
UPI (70%)				
LPI (70%)				

### PSP2.1 Project Plan Summary - Program 8A (Continued)

Student 94419-JUAN ALBERTO GUTIERREZ Date 20160601  
 Program List Sort Program# 8A  
 Instructor JORGE RAFAEL AGUILAR CISI Language C

	Plan	Actual	To Date	To Date%
<b>Defects Injected</b>				
Planning	0.0	1	1	4.0
Design	5.1	3	16	72.0
Design Review	0.0	0	0	0.0
Code	2.0	0	6	24.0
Code Review	0.0	0	0	0.0
Compile	0.0	0	0	0.0
Test	0.0	0	0	0.0
Total Development	7.1	4	25	100.0

<b>Defects Removed</b>				
Planning	0.0	0	0	0.0
Design	0.0	1	1	4.0
Design Review	3.7	2	13	52.0
Code	1.4	1	5	20.0
Code Review	1.4	0	4	16.0
Compile	0.0	0	0	0.0
Test	0.7	0	2	8.0
Total Development	7.1	4	25	100.0
After Development		4	25	

<b>Defect Removal Efficiency</b>			
Def/Hr - DLDR	33.97	4.71	12.40
Def/Hr - CDR	18.31	0.00	5.70
Def/Hr - Compile	0.00	0.00	0.00
Def/Hr - Test	6.43	0.00	2.36
DRL(DLDR/UT)	5.29	1.41	4.16
DRL(CDR/UT)	2.85	0.00	1.91
DRL(Compile/UT)	0.00	0.00	0.00



<b>Test Name/#</b>	1
<b>Test Objective</b>	Ordenar valores de una lista ligada
<b>Test Description</b>	Ordenar los valores de la lista ligada en posición de x
<b>Test Conditions</b>	36 26 8.67 39 7.80 26 8.67 53 7.57 17 5.67 26 8.67 22 7.33 53 8.83 31 10.33 18 6.00 25 8.33 30 7.50 42 14.00 45 9.00 40 10.00 195 39.00 114 38.00 87 21.75 87 29.00 25 8.33 53 13.25 47 11.75 58 5.80 49 7.00 88 14.67 82 16.40 46 5.75 55 13.75 32 10.67 18 6.00 89 29.67 18 6.00 36 12.00 161 40.25 112 37.33 89 22.25
<b>Expected</b>	N/A



# PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

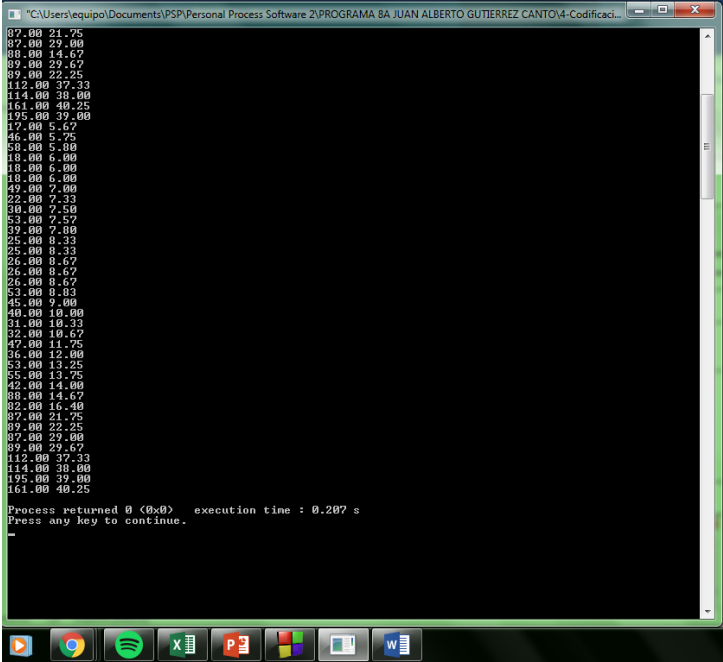
Results	
Actual Results	

Test Name/#	2
Test Objective	Ordenar valores de una lista ligada
Test Description	Ordenar los valores de la lista ligada en posición de y
Test Conditions	36 26 8.67 39 7.80 26 8.67 53 7.57 17 5.67 26 8.67 22 7.33 53 8.83 31 10.33 18 6.00 25 8.33 30 7.50 42 14.00 45 9.00 40 10.00



## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

	195 39.00 114 38.00 87 21.75 87 29.00 25 8.33 53 13.25 47 11.75 58 5.80 49 7.00 88 14.67 82 16.40 46 5.75 55 13.75 32 10.67 18 6.00 89 29.67 18 6.00 36 12.00 161 40.25 112 37.33 89 22.25
<b>Expected Results</b>	N/A
<b>Actual Results</b>	



**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	26/06/2016
Program	List sort	Program #	8A
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><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		



PROCESO PERSONAL DE DESARROLLO DE  
SOFTWARE 2

VERANO 2016

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			
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			
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	
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	26/06/2016
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. <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.				



PROCESO PERSONAL DE DESARROLLO DE  
SOFTWARE 2

VERANO 2016

File Open and Close	Verify that all files are - properly declared - opened - closed	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
<b>Programa 8 A</b>		
<b>No. Del PIP</b>	<b>Descripción del Problema</b>	<b>Descripción de la Propuesta</b>
11	No se sigue el calendario de planeación para el trabajo	Al momento de planear los días que se trabajaran, tener en cuenta las actividades fuera de la clase.

### SIZE ESTIMATING TEMPLATE

Size Estimating Template						
Student		JUAN ALBERTO GUTIERREZ CAN			Date	23/06/2016
Instructor		JORGE RAFAEL AGUILAR CISNEF			Program#	8A
<b>BASE PROGRAM LOC</b>				<b>ESTIMATE</b>	<b>ACTUAL</b>	
BASE SIZE (B) =>				20	30	
LOC DELETED (D) =>				5	5	
LOC MODIFIED (M) =>				10	10	
<b>OBJECT LOC</b>						
BASE ADDITIONS:				TYPE	METHODS	REL. SIZE
PEDIRDATOS()				Data	1	1
main()				Logic	1	VS
(BA) subtotal from page 2				0.0	0	
TOTAL BASE ADDITIONS (BA)				23.9	16	
NEW OBJECTS:				TYPE	METHODS	REL. SIZE
BURBUJALST()				Logic	1	1
(NO) subtotal from page 2				0.0	0	
TOTAL NEW OBJECTS (NO)				23.3	61	
<b>REUSED OBJECTS</b>				LOC	LOC	

REUSED OBJECTS

	LOC	LOC
ISEMPTY()	5.0	6
INSERT()	15.0	20
REMOVE()	10.0	9
Struc nodo()	4.0	4
(R) subtotal from page 2	11.0	6
REUSED TOTAL (R)	45.0	45

		Size	Time
Estimated Object LOC:	$E = BA + NO + M$	57.11	
Regression Parameter:	$B_0$	22.06	0.00
Regression Parameter:	$B_1$	0.88	1.65
Estimated New and Changed LOC:	$N = B_0 + B_1 * E$	72.4	
Estimated Total LOC:	$T = N + B - D - M + R$	122.4	
Estimated Total New Reuse (sum of * LOC):		0	
Estimated Total Development Time:	$Time = B_0 + B_1 * E$		94.3
Prediction Range:	Range	89.2	0.0
Upper Prediction Interval:	$UPI = N + Range$	161.5	
Lower Prediction Interval:	$LPI = N - Range$	0.0	
Prediction Interval Percent		70%	70%
Method Selected		A	C
R^2		0.55	0.00

### Size Estimating Template (continued)

Student  
Instructor

JUAN ALBERTO GUTIERREZ CAN Date  
JORGE RAFAEL AGUILAR CISNEF Program#

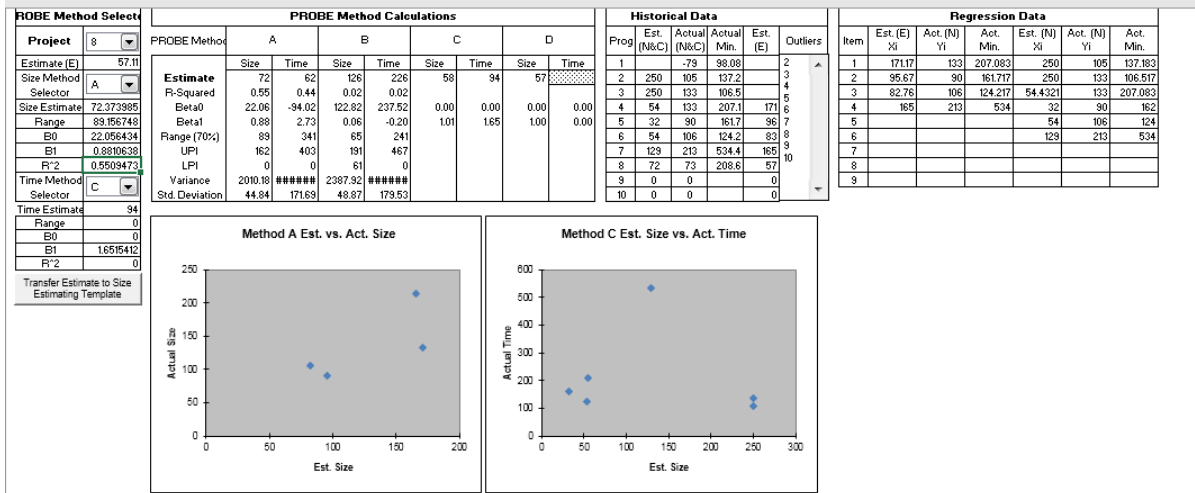
23/06/2016  
8A

[illegible][illegible]



# 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	DLDR	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								

## Operational Specification Template

### Operational Specification Template

Student	Juan Alberto Gutierrez Canto	Date	23/06/2016
Program	List Sort	Program #	8A
Instructor	Jorge Rafael Aguilar Cisneros	Language	C++

Scenario Number	1	User Objective	Dar datos a ordenar
Scenario Objective	Ordenar datos de una lista		



## Functional Specification Template





## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

<b>Student</b>	Juan Alberto Gutierrez Canto	<b>Date</b>	23/06/2016
<b>Program</b>	List Sort	<b>Program #</b>	8A
<b>Instructor</b>	Jorge Rafael Aguilar Cisneros	<b>Language</b>	C++

<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.



## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

Sizeof()	Regresa el tamaño de un dato.
----------	-------------------------------

<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.



## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

### Items

Declaration	Description
Printf()	Imprimir los resultados.

<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.

## State Specification Template

### State Specification Template

Student	Juan Alberto Gutierrez Canto	Date	24/06/2016
Program	List sort	Program #	8A
Instructor	Jorge Rafael Aguilar Cisneros	Language	C++

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;



PROCESO PERSONAL DE DESARROLLO DE  
SOFTWARE 2

VERANO 2016

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	



## Logic Specification Template

### Logic Specification Template

Student	Juan Alberto Gutierrez Canto	Date	25/06/2016
Program	List Sort	Program #	8A
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 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 main

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

**TYPE DEFINITIONS:** Using namespace std;

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

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

Logic reference numbers	Program logic, in pseudocode
1	Mandar a pedir datos
2	Ordenar lista
3	fin

### Time Recording Log.

7	DLD	06/12/16	23:30:23	00:16:09	45.8	TERMINO DE DISEÑO
7	DLDR	06/13/16	14:18:15	14:55:39	37.4	REVISION DE DISEÑO
7	CODE	06/13/16	15:04:09	15:58:58	54.8	CODIFICACION CON LOS DISEÑOS
7	CODE	06/13/16	23:04:49	23:45:14	40.4	COMPLEMENTO DE ESTANDAR DE CODIFICACION Y TERMINO DE CODIFICACION
7	CR	06/14/16	00:51:48	01:17:02	25.2	REVISION DE CODIGO LLENANDO EL TEMPLATE
7	COMPILE	06/14/16	01:21:19	01:22:22	1.1	COMPILACION SIN ERRORES
7	TEST	06/14/16	01:31:50	02:07:47	36.0	TERMINO DE PRUEBAS Y CORRECCION DE ERRORES
7	PM	06/14/16	02:08:24	02:38:11	29.8	LLENADO DE TEMPLATES
8	PLAN	06/23/16	21:14:35	21:35:32	20.9	Planaceacion de programa 8A
8	DLD	06/24/16	22:33:27	00:25:20	111.9	Diseño tomando en cuenta los templates y los diagramas
8	DLDR	06/25/16	12:26:38	12:52:07	25.5	Revisión contemplando el estandar
8	CODE	06/25/16	12:52:18	13:10:41	18.4	Solamente se paso del diseño al código, osea muy facil
8	CR	06/26/16	12:01:11	12:18:02	16.8	Se reviso conforme al estandar
8	COMPILE	06/26/16	12:18:12	12:18:13	0.0	Sin errores la compilación
8	TEST	06/26/16	12:18:21	12:22:38	4.3	Sin errores las pruebas
8	PM	06/26/16	12:23:00	12:33:46	10.8	Completando el proceso

### Defect Recording Log.

Project	Date	Num	Type	Injected	Removed	FixTime	Ref.	Description
7	#####	58	80 DLD	DLDR	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
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
7	#####	60	10 DLD	CODE	1.0			la variable t de el diagrama de clases en la funcion significancia se me olvido borrarla
7	#####	61	10 DLD	CODE	1.0			la funcion CALCULART3 le sobrara una R en el diagrama de clases
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
7	#####	63	80 CODE	CR	3.0			en la linea 187 me faltaba un parentesis en la formula
7	#####	64	80 CODE	CR	1.0			en la funcion correlacion habia un return 0 de mas
7	#####	65	80 CODE	CR	1.0			faltaba un par de parentesis en la funcionEXPX en la formula del valor
7	#####	66	80 CODE	CR	1.0			faltaba agregar la funcion de archivo freopen en el main
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
7	#####	68	80 CODE	TEST	15.0			falla inicializar la regla de simpson por cada iteracion
8	#####	69	10 PLAN	DLD	1.0			Faltaba agregar la funcion struc a el size estimated template.
8	#####	70	20 DLD	DLDR	2.0			Confundi el nombre de la variable Auxiliar con siguiente en el documento de diseño Logic Specification Template
8	#####	71	20 DLD	DLDR	2.0			Me faltaba por la variable booleana llamada ambas en el diagram de clases
8	#####	72	80 DLD	CODE	2.0			Estaba haciendo un ciclo de mas a la hora de imprimir los datos ordenados

### source program listing

\*++++Program+++++

Version: 1.0

Nombre: Juan Alberto Gutierrez Canto



Fecha: 26/06/2016

Descripcion: Programa 8A de la materia proceso personal de software,  
programa que ordena una lista de x,y numeros

+++++  
\*/

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

LOC reusadas:55

LOC modificadas:10

LOC compilación :215

Librerias:

#include <cstdlib>

#include <iostream>

#include<stdlib.h>

#include <stdio.h>

#include <malloc.h>

Clases: N/A

Funciones:

int ISEMPY()

void INSERT(double datox,double datoy)

void REMUEVE()

void PEDIRDATOS()

void BURBUJALTST()

int main()

source: C:\Users\equipo\Documents\proceso personal de software  
2\PROGRAMA 8A JUAN ALBERTO GUTIERREZ CANTO\3-Codificacion

+++++  
\*/

#include <iostream>

#include<stdlib.h>

#include <malloc.h>

#include <stdio.h>

#include <cstdlib>

using namespace std;

bool ambas=false;

long long n;

struct nodo

{ /\*-c estructura para guardar las listas ligadas\*/

double datox,datoy;

struct nodo \*sig;

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

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

```
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;  
}
```

```
/*+++++++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,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->datox = datox;  
        nuevo->datoy=datoy;  
        nuevo-> sig = NULL;  
        raiz = last = nuevo;  
    }  
    else  
    {  
        nuevo = (struct nodo *)malloc( sizeof (struct nodo)); /*-c insertar  
cuando ya existe un nodo*/  
        nuevo->datox = datox;
```

```
nuevo->datoy=datoy;
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()
{
    /*-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*/
    }
}

/*+++++++ seccion
+++++++*/
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 datox,datoy;
    long long i;
    cin>>n;
```



```
for(i=0;i<n;i++)
{
    cin>>datox>>datoy;
    INSERT(datox,datoy);
}
}
```

**/\*+++++++ seccion**  
**+++++++**  
**Function: BURBUJALTST()**  
**Propósito: ordenar una serie de x, y datos**  
**In/out: lista de datos desordenada / lista de datos ordenada**  
**Limitaciones: ninguna**  
**+++++++**  
**+++++++**  
**\*/**

```
void BURBUJALTST()
{
    double datox;
    double datoy;
    long long i;
    long long y;
    struct nodo *inicio=NULL;
    struct nodo *auxiliar=NULL;
    i=0;
    y=0;
    inicio=raiz;
    while(i<n)
    {
        y=i+1;
        auxiliar=inicio->sig;
        while(y<n)
        {
            if(ambas == false)
            {
                if(inicio->datox>auxiliar->datox)
                {
                    datox=auxiliar->datox;
                    datoy=auxiliar->datoy;
                    auxiliar->datox=inicio->datox;
                    auxiliar->datoy=inicio->datoy;
                    inicio->datox=datox;

```

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

}
i=0;
inicio=raiz;
while(i<n)
{
    printf("%.2f %.2f\n",inicio->datox,inicio->datoy);
    inicio=inicio->sig;
    i++;
}
if(ambas==false)
{
    ambas=true;
    BURBUJALTST();
}
}

int main()
{
    freopen("prueba1.txt","r",stdin);
    PEDIRDATOS();
    BURBUJALTST();
    return 0;
}
```

}

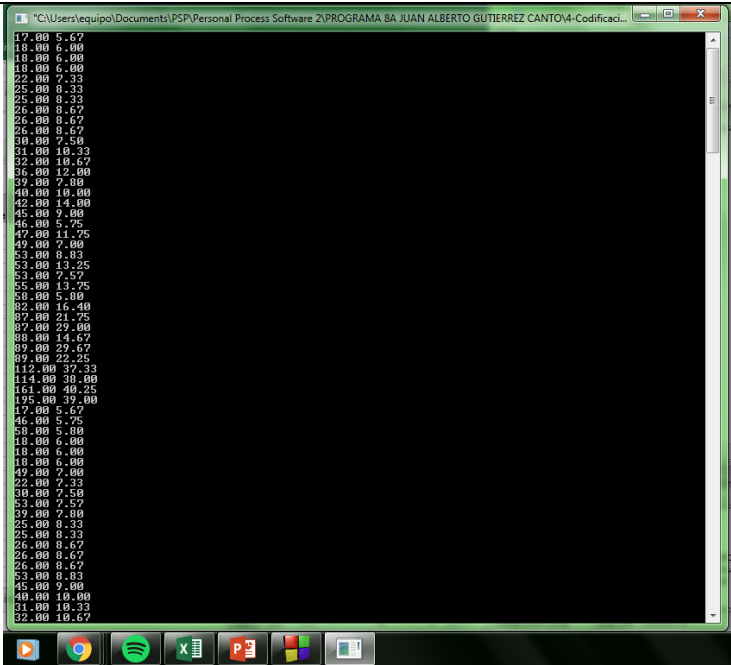
## Test result

<b>Test Name/#</b>	1
<b>Test Objective</b>	Ordenar valores de una lista ligada
<b>Test Description</b>	Ordenar los valores de la lista ligada en posición de x
<b>Test Conditions</b>	36 26 8.67 39 7.80 26 8.67 53 7.57 17 5.67 26 8.67 22 7.33 53 8.83 31 10.33 18 6.00 25 8.33 30 7.50 42 14.00 45 9.00 40 10.00 195 39.00 114 38.00 87 21.75 87 29.00 25 8.33 53 13.25 47 11.75 58 5.80 49 7.00 88 14.67 82 16.40 46 5.75 55 13.75 32 10.67 18 6.00 89 29.67 18 6.00 36 12.00



## PROCESO PERSONAL DE DESARROLLO DE SOFTWARE 2

VERANO 2016

	161 40.25 112 37.33 89 22.25
<b>Expected Results</b>	N/A
<b>Actual Results</b>	

<b>Test Name/#</b>	2
<b>Test Objective</b>	Ordenar valores de una lista ligada
<b>Test Description</b>	Ordenar los valores de la lista ligada en posición de y
<b>Test Conditions</b>	36 26 8.67 39 7.80 26 8.67 53 7.57 17 5.67 26 8.67 22 7.33 53 8.83 31 10.33 18 6.00 25 8.33



	30 7.50 42 14.00 45 9.00 40 10.00 195 39.00 114 38.00 87 21.75 87 29.00 25 8.33 53 13.25 47 11.75 58 5.80 49 7.00 88 14.67 82 16.40 46 5.75 55 13.75 32 10.67 18 6.00 89 29.67 18 6.00 36 12.00 161 40.25 112 37.33 89 22.25
<b>Expected Results</b>	N/A

## Actual Results

```
"C:\Users\equipo\Documents\PSP\Personal Process Software 2\PROGRAMA 8A JUAN ALBERTO GUTIERREZ CANTO\4-Codificaci...
87.00 21.75
87.00 29.00
88.00 14.67
89.00 29.67
89.00 22.25
112.00 37.33
114.00 38.00
161.00 40.25
195.00 39.00
17.00 5.67
46.00 5.75
58.00 5.00
18.00 6.00
18.00 6.00
18.00 6.00
49.00 7.00
22.00 7.33
30.00 7.50
53.00 7.57
39.00 7.00
25.00 8.33
25.00 8.33
26.00 8.67
26.00 8.67
26.00 8.67
53.00 8.83
45.00 9.00
40.00 10.00
47.00 11.75
32.00 10.67
36.00 12.00
53.00 13.25
55.00 13.75
42.00 14.00
88.00 14.67
82.00 16.48
87.00 21.75
89.00 22.25
87.00 29.00
89.00 29.67
112.00 37.33
114.00 38.00
195.00 39.00
161.00 40.25

Process returned 0 (0x0)   execution time : 0.207 s
Press any key to continue.
"
```

Diseño.

