LABORATORIOS DE COMPUTACIÓN SALAS A Y B

Profesor: Marco Antonio Martínez Quintana

Asignatura: Fundamentos de programación

Grupo: 3

Numero de practica: 10

Integrante: Gandarilla Ibarra Jaime

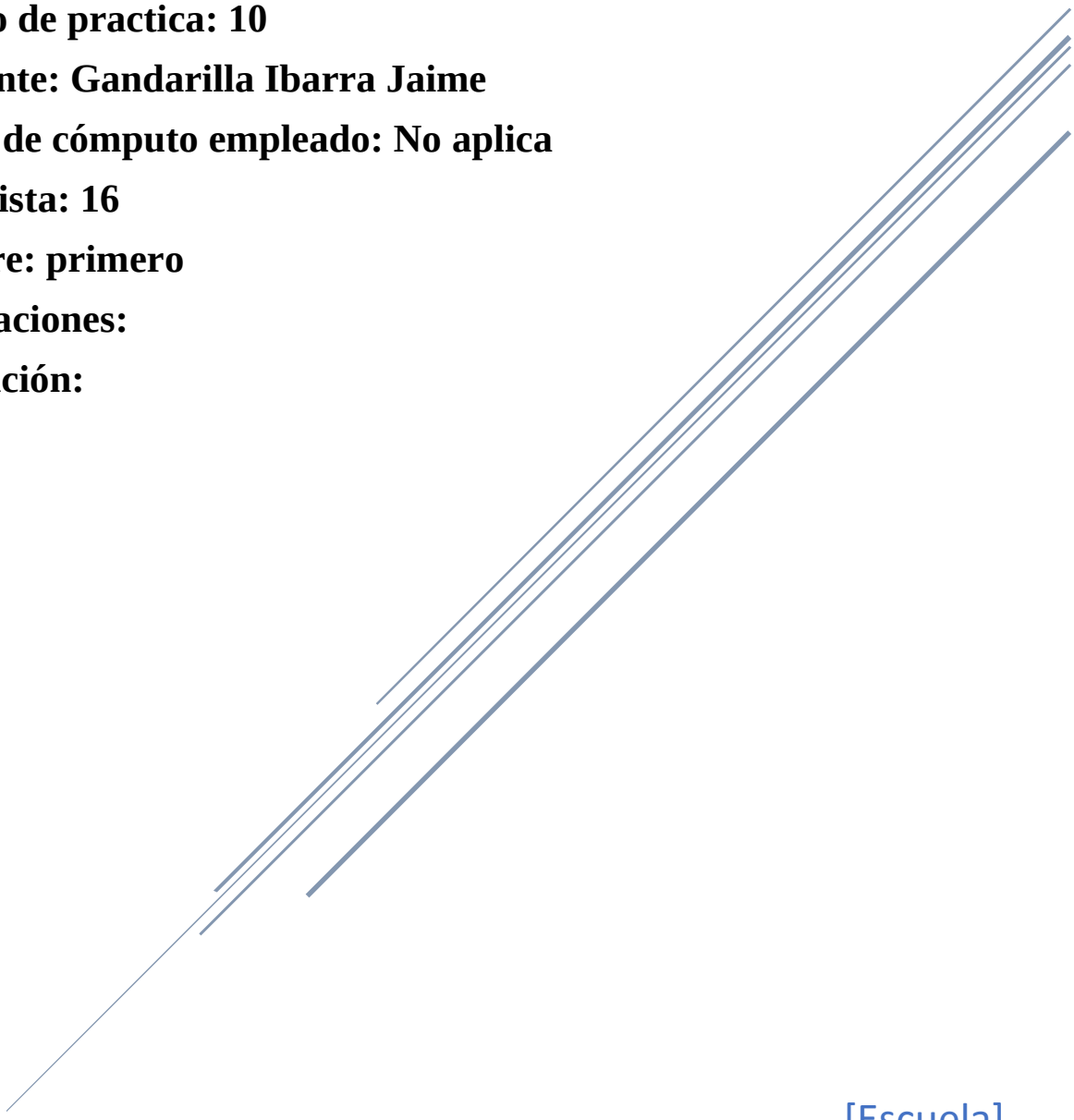
Equipo de cómputo empleado: No aplica

No. de lista: 16

Semestre: primero

Observaciones:

Calificación:



[Escuela]

[Título del curso]

Introducción

Depurar un programa significa someterlo a un ambiente de ejecución controlado por medio de herramientas dedicadas a ello. Este ambiente permite conocer exactamente el flujo de ejecución del programa, el valor que las variables adquieren, la pila de llamadas a funciones, entre otros aspectos. Es importante poder compilar el programa sin errores antes de depurarlo.

Para depurar un programa usando las herramientas desarrolladas por GNU, éste debe compilarse con información para depuración por medio del compilador GCC.

Gcc es un compilador basado en Linux, el cual nos permite compilar en una terminal, de manera que podrá generar programas y por ende habrá que depurarlos de vez en cuando, para esto primero generaremos un ejecutable el cual será el archivo del cual el depurador hará su trabajo. Para esto utilizamos: gcc -g -o Nombre Nombre.c

Posterior a ello iniciamos el depurador con: gdb. / Nombre

Esto nos permitirá abrir el depurador e iniciar una línea de comandos como:

List o l: Permite listar diez líneas del código fuente del programa, si se desea visualizar todo el código fuente debe invocarse varias veces este comando para mostrar de diez en diez líneas. Se puede optar por colocar un número separado por un espacio para indicar a partir de qué línea desea mostrarse el programa. También es posible mostrar un rango de líneas introduciendo el comando y de qué línea a qué línea separadas por una coma. Ejemplo: lista 4,6

b: Establece un punto de ruptura para lo cual debe indicarse en qué línea se desea establecer o bien también acepta el nombre de la función donde se desea realizar dicho paso. Ejemplo: b5 d o delete: Elimina un punto de ruptura, indicando cuál es el que debe eliminarse usando el número de línea. Ejemplo: d 5

clear: Elimina todos los puntos de ruptura. Ejemplo: clear

info line: Permite mostrar información relativa a la línea que se indique después del comando. Ejemplo: infoline 8

run o r: Ejecuta el programa en cuestión. Si el programa tiene un punto de ruptura se ejecutará hasta dicho punto, de lo contrario se ejecutará todo el programa.

c: Continúa con la ejecución del programa después de un punto de ruptura.

s: Continúa con la siguiente instrucción después de un punto de ruptura.

n: Salta hasta la siguiente línea de código después de un punto de ruptura.

p o print: Muestra el valor de una variable, para ello debe escribirse el comando y el nombre de la variable separados por un espacio. Ejemplo: p suma_acumulada

ignore: Ignora un determinado punto de ruptura indicándolo con el número de línea de código.Ejemplo: ignore 5

q o quit: Termina la ejecución de GDB.

Actividad 1

```
#include<stdio.h>

void main()

{

 Int N, CONT, AS;

AS=0;

CONT=1;

printf ("TECLEA UN NUMERO:");

scanf ("%i",&N);

while (CONT<=N)

{

AS=(AS+CONT); CONT=(CONT+2);

}

printf ("\nEL RESULTADO ES %i\n",AS);

}
```

Actividad 2

```
#include<stdio.h>

void main()

{

int i,j;

for(i=1;i<10;i++)

{

printf("\nTabla del%i\n",i);

for(j=1;j==10;j++)

{
```

```
printf("%i X %i = %i\n",i,j,i*j);
}
}
}
```

Actividad 3

```
#include<stdio.h>
void main()
{
int i,j;
for(i=1;i<10;i++)
{
printf("\nTabla del%i\n",i);
for(j=1;j==10;j++)
{
printf("%i X %i = %i\n",i,j,i*j);
}
}
}
```

Actividad 4

```
#include <stdio.h>
#include <math.h>
void main()
{
Int K,X,AP,N;
float AS;
printf("EL TERMINO GENERICO DE LA SERIE ES: X^K/K!");
printf("\n N=");
scanf("%d",&N);
```

```
printf("X=");

scanf("%d",&X);

K=0;

AP=1;

AS=0;

while(K<=N)

{

AS=AS+pow(X,K)/AP;

K=K+1;AP=AP*K;

}

printf("SUM= %ld",AS);

}
```

Conclusiones

La depuración es una herramienta bastante buena al tratarse de los errores en nuestro código ya que basta un paréntesis mal colocado, por ejemplo, o un operador usado de forma incorrecta, para que el programa deje de funcionar como debería. Los depuradores, no solo informan de que se ha encontrado un problema, sino que también dan datos detallados acerca del tipo de error y, a menudo, también indican en qué línea de código se encuentra, y es por eso por lo que es de gran ayuda cuando se trabaja con códigos.