	Caratula para entrega de prácticas
Facultad de ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: Alejandro Pimentel

Asignatura: Fundamentos de Programación

Grupo: 3

No de Práctica(s): 9

Integrante(s): Arteaga Munguía Erick Alejandro

*No. de Equipo de
cómputo empleado:* Peu

No. de Lista o Brigada: 6294

Semestre: 2020-1

Fecha de entrega: 14/10/19

Observaciones:

CALIFICACIÓN: _____

PRACTICA 9

Objetivo:

Elaborar programas en C para la resolución de problemas básicos que incluyan las estructuras de repetición y la directiva define.

¿Para qué sirve la instrucción (do while) en C?

En lenguaje C, para escribir una instrucción while (repetitiva mientras) se utiliza la sintaxis:

```
while ( <expresión_lógica> )  
{  
    <bloque_de_instrucciones>  
}
```

Cuando el <bloque_de_instrucciones> sólo contiene una instrucción, los caracteres *abrir llave* ({) y *cerrar llave* (}) son opcionales.

Por otra parte, al igual que en las instrucciones alternativas doble y simple, a la <expresión_lógica> de una instrucción repetitiva while, también se le llama *condición*.

Para que se ejecute el <bloque_de_instrucciones>, la condición tiene que ser verdadera. Por el contrario, si la condición es falsa, el <bloque_de_instrucciones> no se ejecuta.

Por tanto, cuando el [flujo](#) de un programa llega a un [bucle](#) while, existen dos posibilidades:

1. Si la condición se evalúa a falsa, el bloque de instrucciones no se ejecuta, y el bucle while finaliza sin realizar ninguna [iteración](#).
2. Si la condición se evalúa a verdadera, el bloque de instrucciones sí que se ejecuta y, después, se vuelve a evaluar la condición, para decidir, de nuevo, si el bloque de instrucciones se vuelve a ejecutar o no. Y así sucesivamente, hasta que, la condición sea falsa.

¿Para qué sirve la instrucción (do while) en C?

El ciclo do-while (Instrucción hacer – repetir mientras) es un tipo de estructura repetitiva eficiente. Lo que lo diferencia con el while es que en la estructura do-while la condición se evalúa al finalizar el ciclo, esto hace que las instrucciones **se ejecuten cuando menos una vez**.

La ejecución de esta estructura se realiza de la siguiente manera.

- 1.- Se ejecutan las instrucciones que se encuentran dentro del do, para esto es necesario ponerlas entre llaves.

```
do{  
  
    Instrucciones.  
  
}
```

2.- Después evalúa la expresión dentro de while. Si la expresión es falsa, el ciclo do-while finaliza y pasa a la siguiente instrucción del programa. Si la expresión es verdadera, el ciclo se repite.

La sintaxis de este ciclo es la siguiente

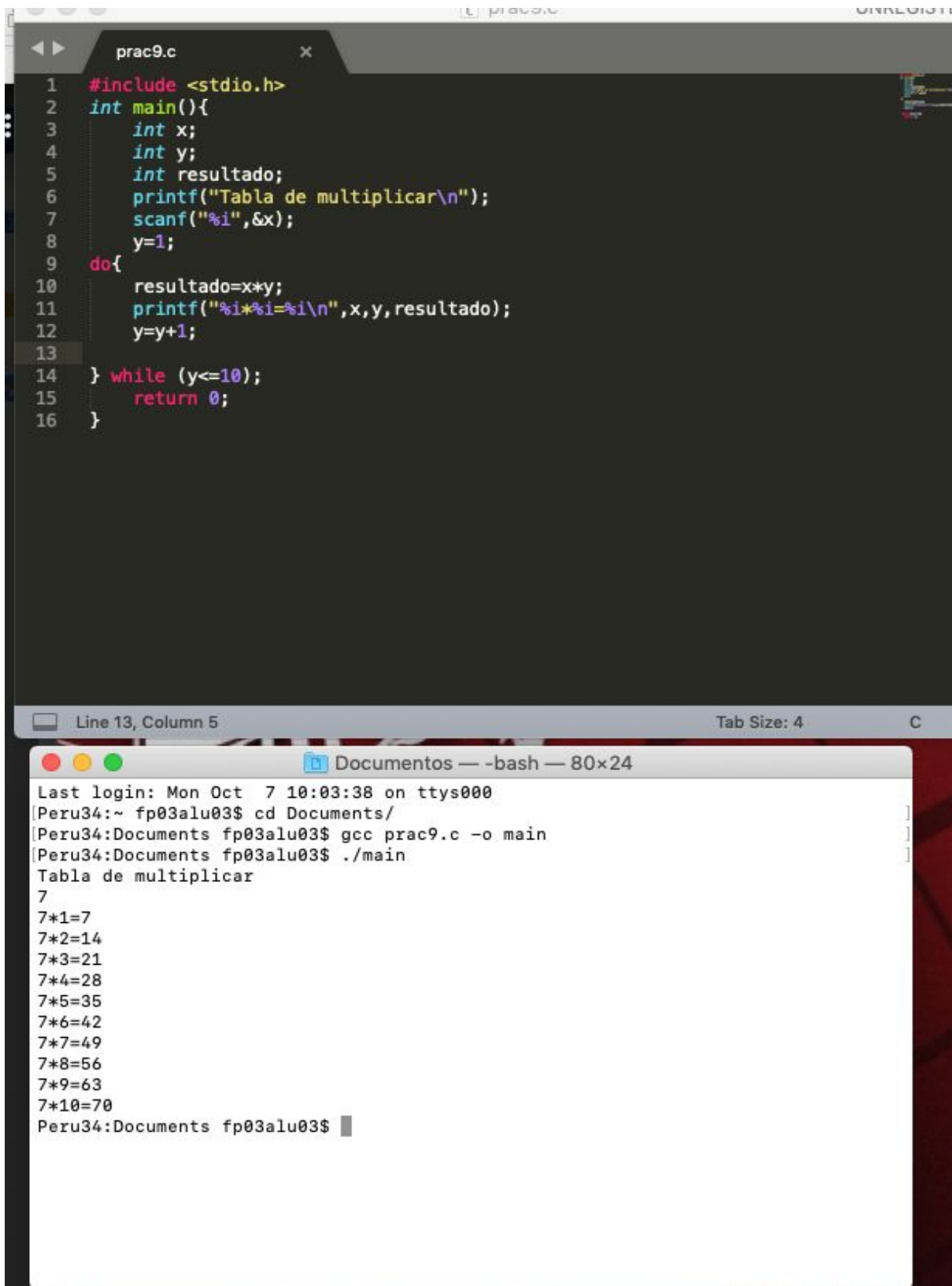
```
do{  
    Instrucción  
}  
while (expresión);
```

¿Para qué sirve la instrucción (for) en C?

Los ciclos for son lo que se conoce como estructuras de control de flujo cíclicas o simplemente estructuras cíclicas, estos ciclos, como su nombre lo sugiere, nos permiten ejecutar una o varias líneas de código de forma iterativa, conociendo un valor específico inicial y otro valor final, además nos permiten determinar el tamaño del paso entre cada "giro" o iteración del ciclo.

En resumen, un ciclo for es una estructura de control iterativa, que nos permite ejecutar de manera repetitiva un bloque de instrucciones, conociendo previamente un valor de inicio, un tamaño de paso y un valor final para el ciclo.

ACTIVIDAD 1: en esta actividad lo que se hizo fue un programa en el que se sacara la tabla de multiplicar.



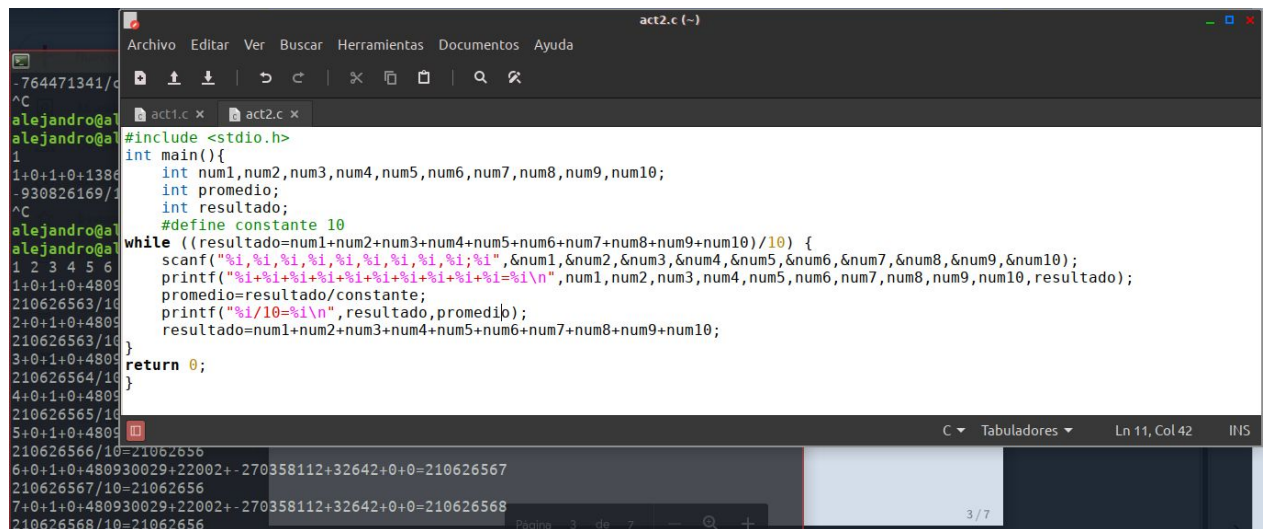
The image shows a code editor window with a file named `prac9.c` and a terminal window below it. The code in `prac9.c` is a C program that prints a multiplication table for a user-defined number `x` (ranging from 1 to 10). The terminal shows the compilation and execution of this program, resulting in the output of the multiplication table for `x=7`.

```
1  #include <stdio.h>
2  int main(){
3      int x;
4      int y;
5      int resultado;
6      printf("Tabla de multiplicar\n");
7      scanf("%i",&x);
8      y=1;
9      do{
10         resultado=x*y;
11         printf("%i*%i=%i\n",x,y,resultado);
12         y=y+1;
13     } while (y<=10);
14     return 0;
15 }
16 }
```

Line 13, Column 5 Tab Size: 4 C

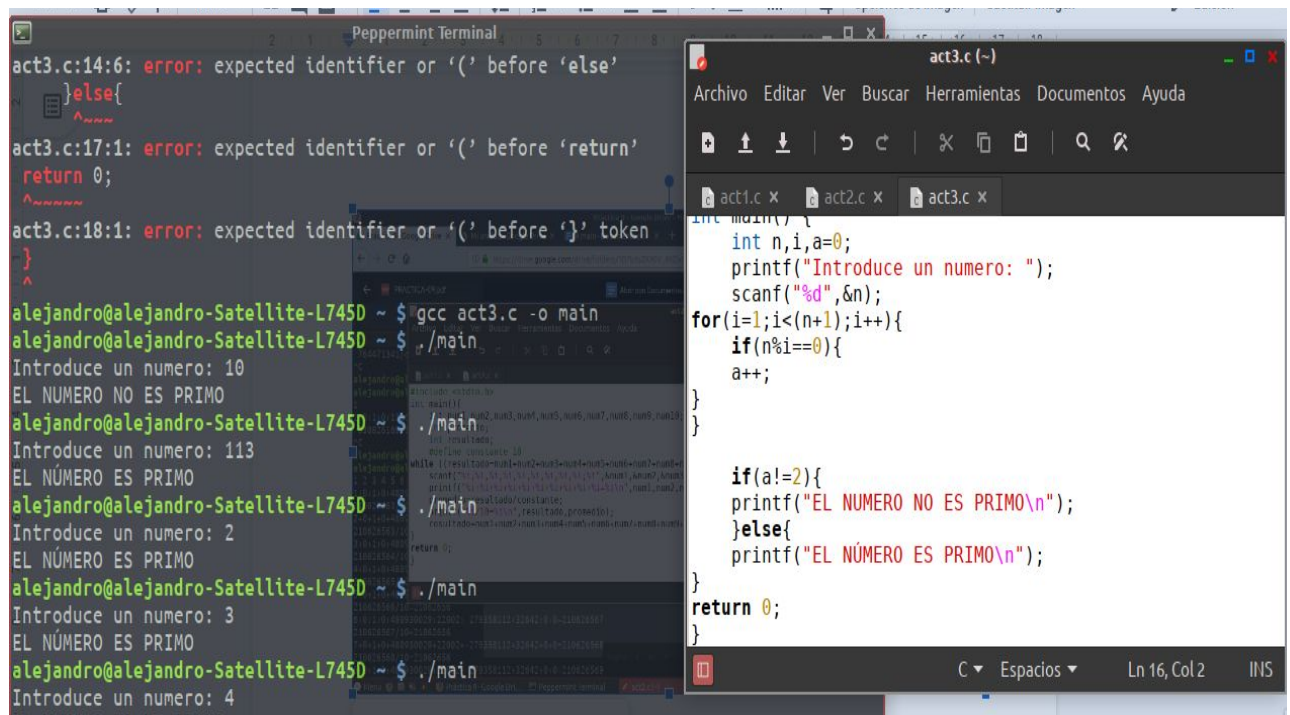
```
Documents — -bash — 80x24
Last login: Mon Oct 7 10:03:38 on ttys000
[Peru34:~ fp03alu03$ cd Documents/
[Peru34:Documents fp03alu03$ gcc prac9.c -o main
[Peru34:Documents fp03alu03$ ./main
Tabla de multiplicar
7
7*1=7
7*2=14
7*3=21
7*4=28
7*5=35
7*6=42
7*7=49
7*8=56
7*9=63
7*10=70
Peru34:Documents fp03alu03$
```

ACTIVIDAD 2: se hizo la suma y el promedio de 10 números



```
act2.c (~)
Archivo Editar Ver Buscar Herramientas Documentos Ayuda
act1.c x act2.c x
#include <stdio.h>
int main(){
    int num1,num2,num3,num4,num5,num6,num7,num8,num9,num10;
    int promedio;
    int resultado;
    #define constante 10
    while ((resultado=num1+num2+num3+num4+num5+num6+num7+num8+num9+num10)/10) {
        scanf("%i%i%i%i%i%i%i%i%i%i", &num1,&num2,&num3,&num4,&num5,&num6,&num7,&num8,&num9,&num10);
        printf("%i+i+i+i+i+i+i+i+i+i=%i\n", num1,num2,num3,num4,num5,num6,num7,num8,num9,num10,resultado);
        promedio=resultado/constante;
        printf("%i/10=%i\n",resultado,promedio);
        resultado=num1+num2+num3+num4+num5+num6+num7+num8+num9+num10;
    }
    return 0;
}
```

ACTIVIDAD 3: En este último programa se hizo que leyera un número y diga si es primo o no lo es.



```
act3.c:14:6: error: expected identifier or '(' before 'else'
    }else{
    ^
act3.c:17:1: error: expected identifier or '(' before 'return'
    return 0;
    ^
act3.c:18:1: error: expected identifier or '(' before '}' token
}
^
alejandro@alejandro-Satellite-L745D ~ $ gcc act3.c -o main
alejandro@alejandro-Satellite-L745D ~ $ ./main
Introduce un numero: 10
EL NUMERO NO ES PRIMO
alejandro@alejandro-Satellite-L745D ~ $ ./main
Introduce un numero: 113
EL NÚMERO ES PRIMO
alejandro@alejandro-Satellite-L745D ~ $ ./main
Introduce un numero: 2
EL NÚMERO ES PRIMO
alejandro@alejandro-Satellite-L745D ~ $ ./main
Introduce un numero: 3
EL NÚMERO ES PRIMO
alejandro@alejandro-Satellite-L745D ~ $ ./main
Introduce un numero: 4
EL NUMERO NO ES PRIMO

act3.c (~)
Archivo Editar Ver Buscar Herramientas Documentos Ayuda
act1.c x act2.c x act3.c x
int main(){
    int n,i,a=0;
    printf("Introduce un numero: ");
    scanf("%d",&n);
    for(i=1;i<(n+1);i++){
        if(n%i==0){
            a++;
        }
    }
    if(a!=2){
        printf("EL NUMERO NO ES PRIMO\n");
    }else{
        printf("EL NÚMERO ES PRIMO\n");
    }
    return 0;
}
```

CONCLUSIÓN:

Me pareció una práctica buena ya que vimos los diferentes tipos de ciclos y los empleamos en los diversos programas que realizamos, en lo personal se me facilitó más el ciclo do while (fue el primero que hice), y el que más se me complicó fue el while.

Pero fuera de eso me gusto la práctica.