|  |  |  |
| --- | --- | --- |
|  | **Carátula para entrega de prácticas** | |
| Facultad de Ingeniería | | Laboratorio de docencia |

Laboratorios de computación

salas A y B

|  |  |
| --- | --- |
| *Profesor:* | Claudia Rodriguez Espino |
| *Asignatura:* | Fundamentos de programación |
| *Grupo:* | 4 |
| *No de Práctica(s):* | 9 |
| *Integrante(s):* | Garduño Pérez Angel Isaac |
|  |  |
| *No. de Equipo de cómputo empleado* | 16 |
| *Semestre:* | 2 |
| *Fecha de entrega:* | 05/04/2019 |
| *Obervaciones:* |  |
|  |  |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_\_

**Guía de práctica de estudio 09:**

**Estructuras de repetición**

**Objetivo:**

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

**Actividades:**

x Elaborar un programa que utilice la estructura while en la solución de un problema

x Elaborar un programa que requiera el uso de la estructura do-while para resolver un problema.

x Hacer la comparación con el programa anterior para distinguir las diferencias de operación entre while y do-while.

x Resolver un problema dado por el profesor que utilice la estructura for en lugar de la estructura while.

x Usar la directiva define para elaboración de código versátil.

**Estructura de control repetitiva while**

La estructura repetitiva (o iterativa) while primero valida la expresión lógica y si ésta se cumple (es verdadera) procede a ejecutar el bloque de instrucciones de la estructura, el cual está delimitado por las llaves {}. Si la condición no se cumple se continúa el flujo normal del programa sin ejecutar el bloque de la estructura, es decir, el bloque se puede ejecutar de cero a ene veces. Su sintaxis es la siguiente:

while (expresión\_lógica) {

// Bloque de código a repetir

// mientras que la expresión

// lógica sea verdadera.

}

Si el bloque de código a repetir consta de una sola sentencia, entonces se pueden omitir las llaves.

**Estructura de control repetitiva do-while**

do-while es una estructura cíclica que ejecuta el bloque de código que se encuentra dentro de las llaves y después valida la condición, es decir, el bloque de código se ejecuta de una a ene veces. Su sintaxis es la siguiente:

do {

/\* Bloque de código que se ejecuta por lo menos una vez y se repite mientras la expresión lógica sea verdadera. \*/

} while (expresión\_lógica);

Si el bloque de código a repetir consta de una sola sentencia, entonces se pueden omitir las llaves. Esta estructura de control siempre termina con el signo de puntuación ';'.

**Estructura de control de repetición for**

Lenguaje C posee la estructura de repetición for la cual permite realizar repeticiones cuando se conoce el número de elementos que se quiere recorrer. La sintaxis que generalmente se usa es la siguiente:

for (inicialización ; expresión\_lógica ; operaciones por iteración) {

/\* Bloque de código a ejecutar \*/

}

La estructura for ejecuta 3 acciones básicas antes o después de ejecutar el bloque de código. La primera acción es la inicialización, en la cual se pueden definir variables e inicializar sus valores; esta parte solo se ejecuta una vez cuando se ingresa al ciclo y es opcional. La segunda acción consta de una expresión lógica, la cual se evalúa y, si ésta es verdadera, ejecuta el bloque de código, si no se cumple se continúa la ejecución del programa; esta parte es opcional. La tercera parte consta de un conjunto de operaciones que se realizan cada vez que termina de ejecutarse el bloque de código y antes de volver a validar la expresión lógica; esta parte también es opcional.

**Define**

Las líneas de código que empiezan con # son directivas del preprocesador, el cual se encarga de realizar modificaciones en el texto del código fuente, como reemplazar un símbolo definido con #define por un parámetro o texto, o incluir un archivo en otro archivo con #include. define permite definir constantes o literales; se les nombra también como constantes simbólicas. Su sintaxis es la siguiente:

#define <nombre> <valor>

Al definir la constante simbólica con #define, se emplea un nombre y un valor. Cada vez que aparezca el nombre en el programa se cambiará por el valor definido. El valor puede ser numérico o puede ser texto.

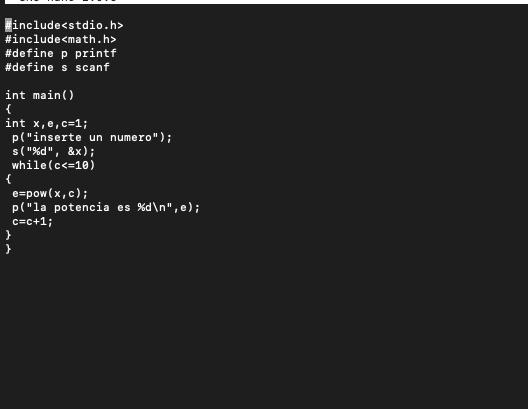
**Break**

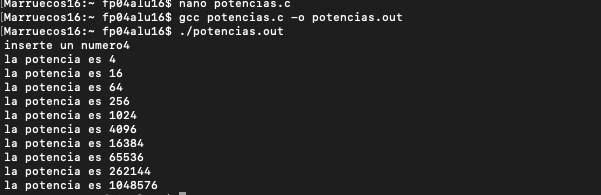
Algunas veces es conveniente tener la posibilidad de abandonar un ciclo. La proposición break proporciona una salida anticipada dentro de una estructura de repetición, tal como lo hace en un switch. Un break provoca que el ciclo que lo encierra termine inmediatamente.

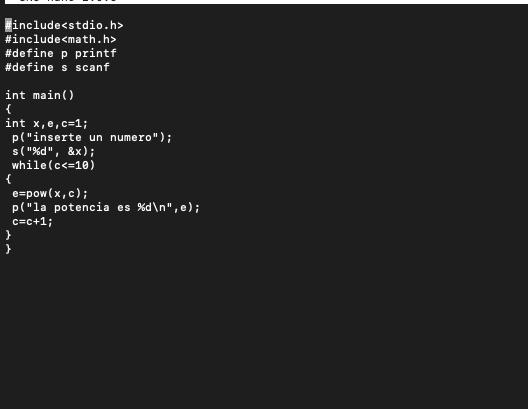
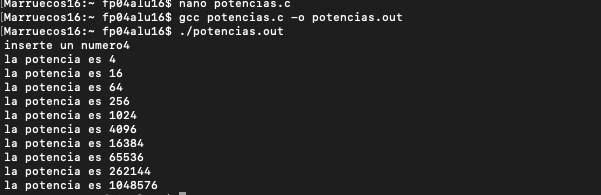
**Continue**

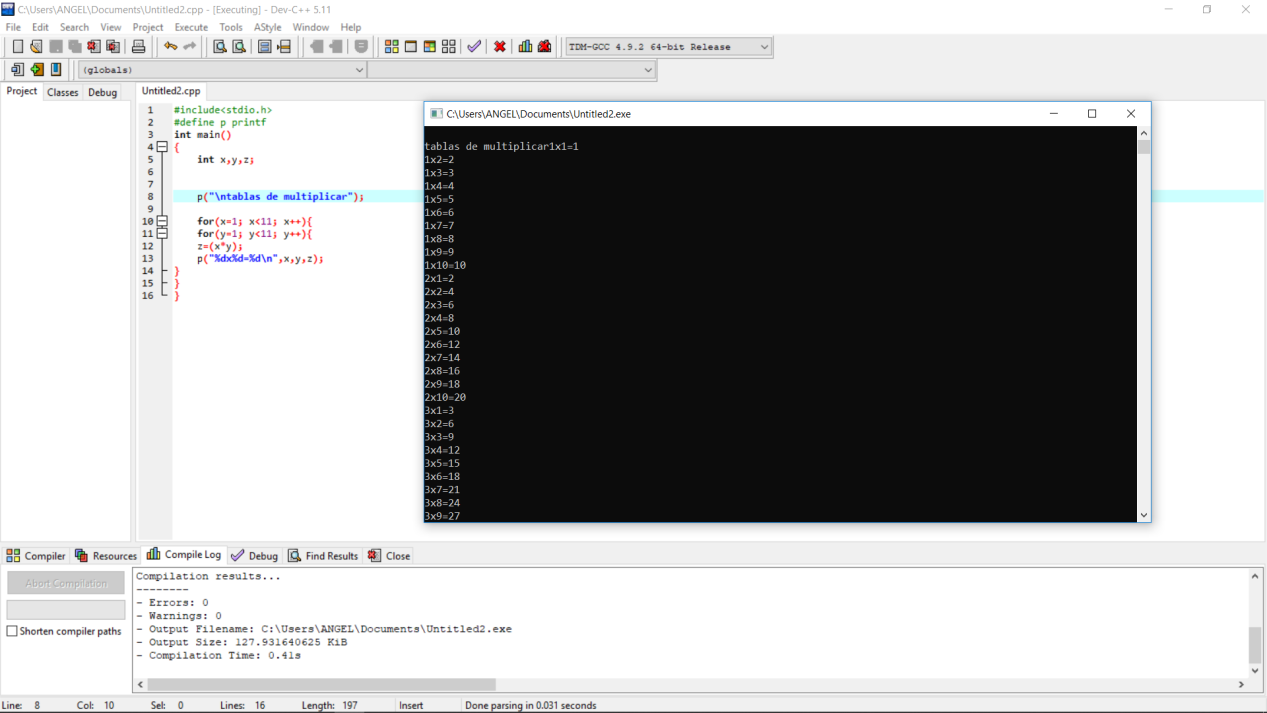
La proposición continue provoca que inicie la siguiente iteración del ciclo de repetición que la contiene.

Actividades:









CONCLUSIONES

Gracias a las estructuras de repetición nos ahorramos mucho tiempo programando porque nos permite ejecutar una sola función varias veces hasta que sea necesario sin tener que escribir esa misma función mas de una vez en el código o sin declarar muchas variables.