|  |
| --- |
| Laboratorio de Computación  Salas A y B |
| |  |  | | --- | --- | | Profesor: | Claudia Rodriguez Espino | | Asignatura: | Fundamentos de Programación | | Grupo: | 1104 | | No de Práctica(s): | 1 | | Integrante(s): | Valencia Mancera Erick Samuel | |  |  | |  |  | |  |  | | No. de Equipo de cómputo empleado: | #52 | | Semestre: | 1ª | | Fecha de entrega: | 08 de octubre de 2018 | | Observaciones: |  | |  |  | |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Práctica #9: Estructuras de Repetición**

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

**Desarrollo**

Las estructuras de repetición son las llamadas estructuras cíclicas, iterativas o de bucles. Permiten ejecutar un conjunto de instrucciones de manera repetida (o cíclica) mientras que la expresión lógica a evaluar se cumpla (sea verdadera). En lenguaje C existen tres estructuras de repetición: while, do-while y for. Las estructuras while y do-while son estructuras repetitivas de propósito general.

*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 n veces.

*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.

*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.

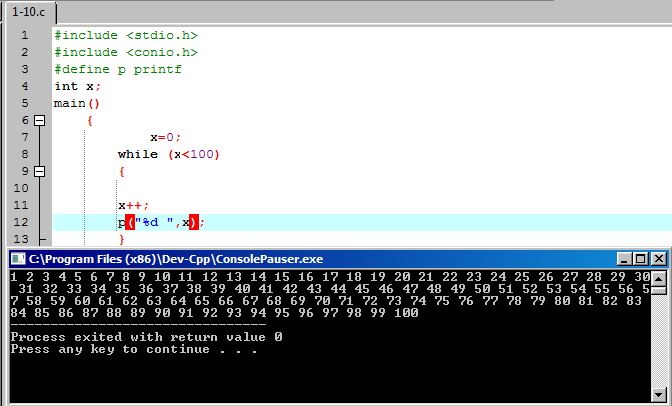
La estructura for ejecuta 3 acciones básicas antes o después de ejecutar el bloque de código. La primera acción es la inicialización, en la cual se pueden definir variables e inicializar sus valores; esta parte solo se ejecuta una vez cuando se ingresa al ciclo y es opcional. La segunda acción consta 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*

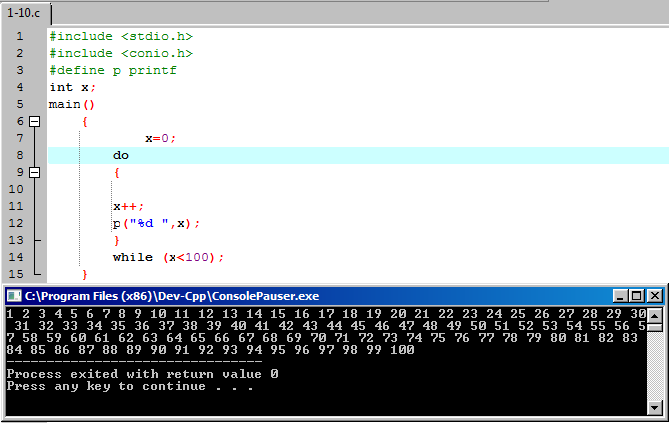
Las líneas de código que empiezan con # son directivas del preprocesador, el cual se encarga de realizar modificaciones en el texto del código fuente, como reemplazar un símbolo definido con #define por un parámetro o texto, o incluir un archivo en otro archivo con #include. *define* permite definir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

**Actividades**

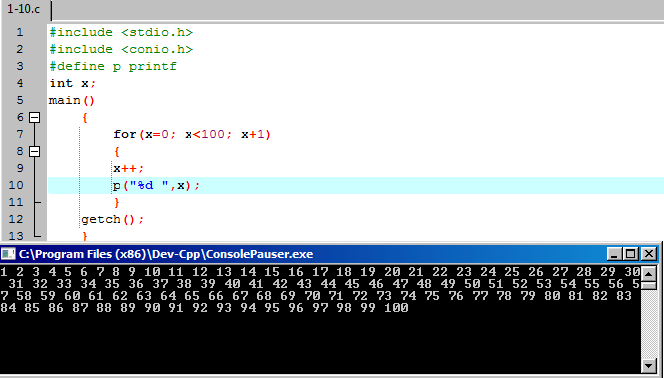
**Actividad 1:** Elaborar programa que imprima los números del 1-100 con los ciclos “for”, “while” y “do-while”



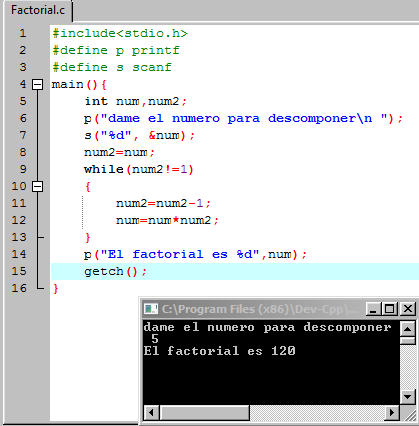
**While**



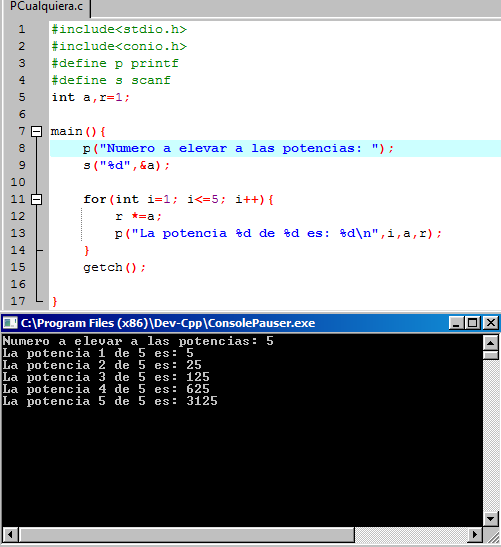
**Do -while**



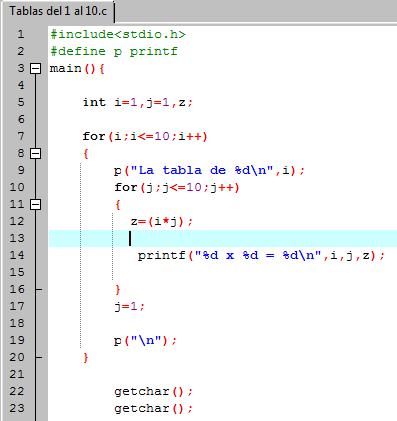
**For**

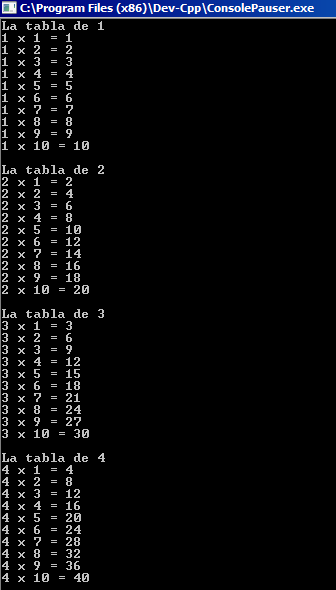
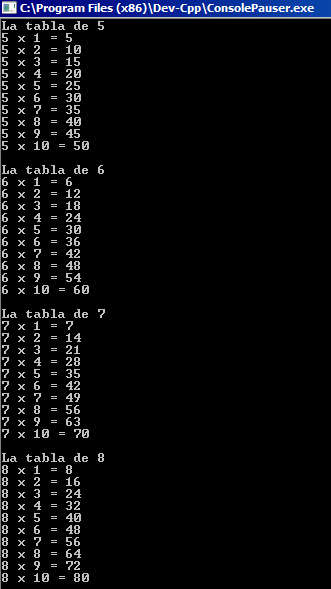
**Actividad 2:** Elabora el programa que calcule factorial de un número.

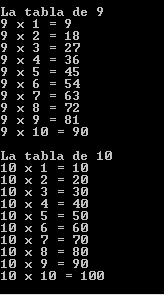
Actividad 3: Que imprima las primeras cinco potencias de cualquier número del 1 al 5.



Actividad 4: Elabora programa que imprima las todas las tablas del 1-10 x10.





**Conclusión**

Al estar programando en lenguaje C++, debemos tener en cuenta que podemos utilizar otras formas de comandos para hacer menor la sintaxis del programa en sí. Esto ha servido para reforzar los conocimientos de programación que hemos obtenido a lo largo del semestre.