Universidad Autónoma de Baja California

Ingeniería en Software y Tecnologías Emergentes.

Taller 5. Clases de almacenamientos

Materia: Lenguaje C

Maestro: Yulith Altamirano

Alumno: Diego Quiros 372688

Fecha: 09/30/2023

Instrucciones:

Durante este taller, vamos a resolver ejercicios relacionados con la unidad 3. Para cada ejercicio, declaren las variables, constantes y funciones necesarias para llevar a cabo la tarea requerida. *Para los ejercicios con * se invocaran 10 veces las funciones.

Ejercicios:

1. Declara una variable automática llamada contador en una función. Incrementa su valor en un bucle y muestra su valor en cada iteración.*

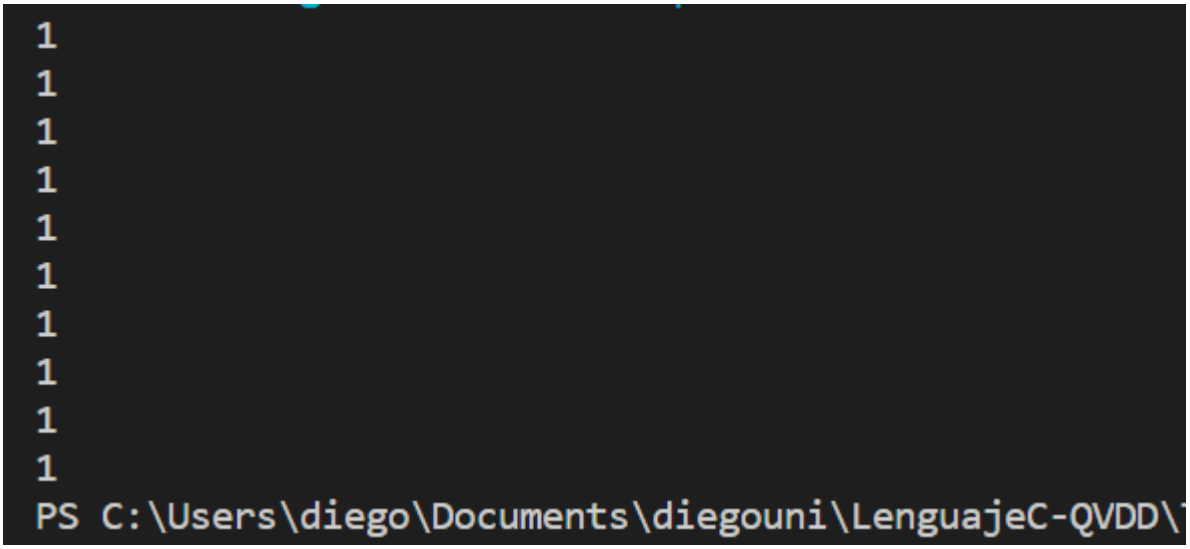
Código:

```C/C++
#include <stdio.h>

void contadorAutomatico(void)
{
  //Declaracion de la variable automatica
  auto int contador = 0;
  contador++;
  printf("%d\n",contador);
}

//La variable contador deja de existir al salir de la funcion

int main()
{
  for(int i = 0 ; i < 10 ; i++)
  {
    contadorAutomatico();
  }
  //Fuera de la funcion la variable ya no es accesible
  return 0;
}
```

¿Qué sucede con la variable al salir de la función?

Al salir de la función la variable muere, su vida útil esta limitada a cuando entra a la función por lo que en la salida mostrará solamente el valor que se le otorga a dentro de la función es decir “1”.

Salida:



2. Declara una variable externa llamada saldo en un archivo fuente (archivo.c) y accede a ella desde otro archivo fuente (otroarchivo.c). Modifica su valor en ambos archivos y muestra el valor final.
Código Archivo A:

```C/C++
//Archivo A
#include <stdio.h>
//Declarando la variable externa
extern int saldo=100;
```

Código Archivo B:

```C/C++
//Archivo B
#include <stdio.h>

//Declarando la variable externa
int main()
{
  extern int saldo;
  //Asignacion de valor a la variable Saldo
  printf("Valor modificado del saldo en archivo A %d\n",saldo);
  saldo=50;
  printf("Valor modificado del saldo en archivo B %d\n",saldo);
  return 0;
}
```

¿Cómo afecta la visibilidad y el tiempo de vida de la variable externa?

La variable externa se puede mostrar en todo el programa e igualmente su tiempo de vida puede durar lo que dure la ejecución del programa.

Como se manda a llamar:

```
diego@Dieguin MINGW64 ~/Documents/diegouni/LenguajeC-QVDD/Talleres/Taller5_ClasesDeAlmacenamiento
(main)
$ gcc QVDD_LenguajeC_taller5_ejercicio2A.c QVDD_LenguajeC_taller5_ejercicio2B.c -o programamain
```

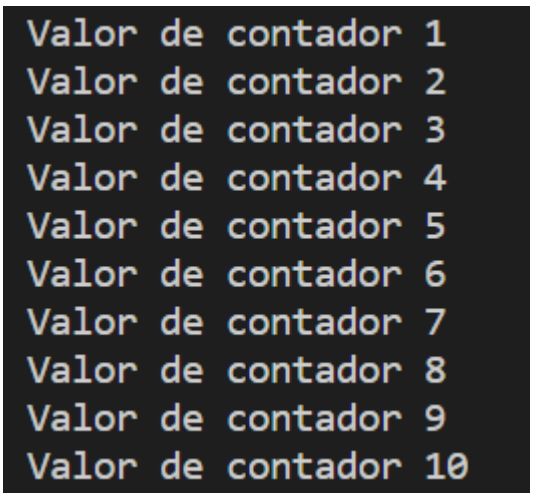
Salida:

```
$ ./programamain
Valor modificado del saldo en archivo A 100
Valor modificado del saldo en archivo B 50
```

3. Declara una variable estática llamada contador en una función y muestra su valor en cada llamada a la función. *

```c
C/C++
#include <stdio.h>

void funcioncontador(void)
{
  //Declaracion de variable estatica
  static int contador = 0;
  contador++;
  //mostrar el valor del contador
  printf("Valor de contador %d\n",contador);
}

//La variable contador deja de existir al salir de la funcion

int main()
{
  for (int i = 0; i < 10; i++)
  {
    funcioncontador();
  }

  return 0;
}
```

¿Qué sucede con la variable al salir de la función? ¿Cómo difiere de una variable automática?
Al salir de la función la variable mantiene el valor que obtuvo, a diferencia de la variable automática la cual pierde su contenido al momento de salir de la función.
Salida:



4. Declara una variable de registro llamada temp y otra automática llamada valor en una función. Compara el acceso y el tiempo de vida de estas variables.

```C/C++
#include <stdio.h>

int main()
{
  // Declaración de una variable automática
  auto int valor = 42;

  // Declaración de una variable de registro
  register int temp = 0;

  // Acceso y modificación de la variable automática
  valor = valor + 10;

  // Acceso y modificación de la variable de registro
  temp = valor + 5;

  printf("Valor de la variable automática: %d\n", valor);
  printf("Valor de la variable de registro: %d\n", temp);

  return 0;
}
```

¿Por qué usarías una variable de registro en lugar de una variable automática?
Se usará una variable de registro para acceder al espacio de memoria con más velocidad, sin embargo al tener un límite los registros dependiendo la arquitectura de cada procesador, los compiladores más recientes tienen el poder de rechazar la instrucción.

5. Declara una variable global llamada pi con un valor de 3.14159 y otra variable local con el mismo nombre en una función (Con diferente valor). Intenta acceder a ambas variables desde diferentes partes del programa.

```C/C++
#include <stdio.h>

//declaracion de variable global
float pi = 3.1416;

void funcion(void)
{
  //variable local con el mismo nombre
  float pi = 3.14;

  printf("Valor de pi dentro de la funcion: %f",pi);
}

int main()
{
  //Llamamos a la variable global pi
  printf("Valor de pi Global: %f\n",pi);

  //Llamada a la guncion que tiene su propia funcion pi
  funcion();
  //Volvemos a acceder a la variable global
  printf("\nValor de pi Global despues de llamar a la funcion: %f\n",pi);

  return 0;
}
```

¿Cuál es el resultado? Explica el concepto de ámbito y visibilidad.
Ámbito se refiere a la parte del programa donde la variable es válida y accesible y visibilidad se refiere a la capacidad del código de acceder a una variable.
Ejemplo: