

Práctico Nº 5

Tema: Estructuras en C

Nota:

- Todos los programas deben ser codificados usando el IDE Eclipse. Crear un espacio de trabajo (workspace) Practico5 y para cada ejercicio crear en dicho espacio de trabajo un proyecto Pr5_ejxx. En A partir del Ej.12 realizar los programas utilizando **funciones**.

Ej. 1- Explicar en qué consiste un IDE y cuáles son sus componentes. Mencionar al menos 2 ejemplos de IDE. Explicar el concepto de **workspace** en Eclipse.

Ej. 2- Visualizar los videos “*Mi primer proyecto C con Eclipse (1/2)*” y “*Mi primer proyecto C con Eclipse (2/2)*”, disponibles en la pestaña Proyecto de Laboratorio del AV.

Ej. 3- Abrir la aplicación Eclipse y explorar el entorno de trabajo. Indicar los nombres que recibe cada colección de paneles o vistas. Analizar la vista *Explorador de Proyectos*.

Ej. 4- Explicar en qué consiste un proyecto para Eclipse. En el workspace Practico5 de su home crear un proyecto de C llamado **Pr5_Ej4** de tipo ejecutable “*Hello Word ANSI C Project*” y usando el compilador “Linux GCC”. Analizar qué cambios se producen en el Explorador de Proyectos, en el Editor y en la Vista Esquema.

Ej. 5- Modificar el código fuente creado en el ejercicio anterior, reemplazando la instrucción puts por printf. Además, la función main debe retornar el valor 0. Guardar los cambios y compilar el proyecto. Analizar qué modificaciones se produjeron en el *Explorador de Proyectos*. Ejecutar.

Ej. 6- Crear un proyecto vacío (*Empty Project*), denominado **Prac5_ej6**. Realizar las siguientes acciones:

1. Descargar del aula virtual el archivo **pr5_ej6.c** y almacenarlo en el directorio del proyecto recién creado.
2. Renovar (F5) en Eclipse (*Explorador de Proyectos-Botón derecho en el proyecto Pr5_ej6-Renovar*).
3. Editar el archivo **pr5_ej6.c**.
4. Compilar (*Proyecto >> Construir Proyecto*), y analizar los errores de sintaxis. Corregir los errores hasta que se construya el ejecutable.

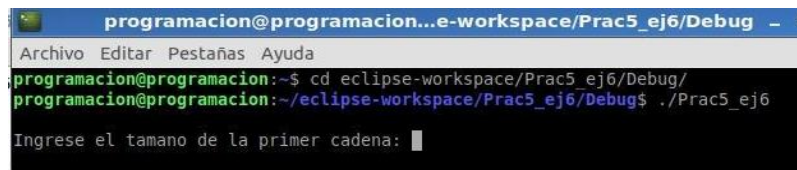
Nota: *el ejecutable es creado cuando el programa no tiene errores de sintaxis, pero puede tener avisos (warnings) que conviene revisar.*

5. Ejecutar (*Ejecutar como >> 2 Local C/C++ Aplicación*). ¿Funciona correctamente este programa?
6. ¿Cuál es la opción para depurar un programa? Iniciar la depuración (*Depurar como >> 2 Local C/C++ Aplicación*) ¿Se produce un cambio de perspectiva? Analizar todos los paneles especialmente las vistas de *Variables* y *Puntos de Interrupción (breakpoints)*.
7. Explicar la diferencia entre ejecutar y depurar.
8. ¿Qué significa colocar un punto de interrupción o Breakpoint en un programa? Explicar cómo se colocan y como se eliminan.
9. Colocar un breakpoint en la línea 42 (printf (“\n Ingrese tamaño.....)) de la función main. ¿Cómo se visualiza la colocación de un breakpoint?
10. Colocar otro breakpoint en la línea 53 (printf (“\n Ingrese tamaño.....)).
11. Reanudar el depurador (<F8> o *Ejecutar - Reanudar*).
12. Indicar donde se detuvo ¿se visualizan todas las variables locales del main? Si alguna no se visualiza, agregarla (en el panel *Variables*, botón derecho – *Crear expresión de observación*). Otra manera de visualizar una variable es seleccionarla en el código y con botón derecho seleccionar *Add Watch Expression*.
13. Continuar con la depuración – Oprimir <F6> avanza línea por línea. O <F8> para avanzar al próximo Breakpoint.
14. Ingresar los datos solicitados en la consola de ejecución.
15. Cuando se detenga en la línea 53 (breakpoint), en la ventana examinador, visualizar si las variables involucradas hasta ese punto se han modificado correctamente.
16. Si todo es correcto, continuar ejecutando línea a línea, presionando <F6>, hasta la línea 62 (printf (“\n Ingrese la posición...)).

17. Posteriormente, observar en el panel “Variables” si hasta ese punto, se han modificado correctamente.
18. Observar si Cad2 tiene almacenado lo que se acaba de ingresar por teclado. Deducir porque Cad2 tiene esos valores.
19. Detener el depurador. (Control + F2).
20. Realizar las correcciones necesarias en el código fuente, guardar los cambios y compilar nuevamente.
21. Ejecutar. Responder si ahora funciona correctamente.
22. Agregar un Breakpoint en la línea 62 (*printf("\nIngrese la posición*).
23. Reiniciar la depuración y analizar los valores de las variables cada vez que se detenga en un breakpoint.
24. Responder si funciona correctamente.
25. Eliminar los breakpoints de las líneas 42 y 53 (*Oprimir el botón rojo de dicha línea*).
26. Comenzar nuevamente la depuración.

Si fuera necesario, agregar las variables que se necesiten en el examinador. Continuar depurando y solucionar todos los problemas. Si se desea seguir paso a paso la depuración de una función oprimir <F5> (*ingreso al código de la función*).

Nota: Una vez que el programa sea depurado, se recomienda **compilar** el programa desde el IDE Eclipse (*Construir proyecto*) y **ejecutarlo** desde una terminal (*LXTerminal*). Para ello, se debe abrir una consola o terminal y dirigirse al directorio donde se encuentra el archivo ejecutable del proyecto. Una vez allí, lo ejecutamos con el comando `./<archivo-ejecutable>` como se muestra en la imagen.



Ej. 7- Descargar del aula virtual el archivo el archivo *pr5_Ej7_depurar.c*. Crear un proyecto con dicho código fuente, compilar y ejecutar. En caso de que el funcionamiento no sea el esperado, utilizar el depurador para corregirlo, analizando en cada sentencia el valor que toman las variables involucradas.

Ej. 8– Dado el siguiente código:

```
# include <stdio.h>
int main (){
    float real = 3.5, *p_real;
    int ent = 4356, *p_ent;
    char car = 'A', *p_car;
    char let[5]={'A','E','I','O','U'}, *p_let;
    int num[3]= {111,-23,66}, *p_num;

    printf("El tamaño de un real es %d y del puntero es %d \n",
           sizeof(real), sizeof(p_real));
    printf("El tamaño de un entero es %d y del puntero es %d\n",
           sizeof(ent), sizeof(p_ent));
    printf("El tamaño de un caracteres %d y del puntero es %d\n",
           sizeof(car), sizeof(p_car));
    printf("El tamaño del arreglo es %d y del puntero es %d\n",
           sizeof(let), sizeof(p_let));

    p_let = let;
    printf("El tamaño de una posición de arreglo es %d y del puntero es %d\n",
           sizeof(let[0]), sizeof(p_let));

    p_num = num;
    printf("El tamaño del arreglo es %d y su dirección es %d\n",
           sizeof(num), sizeof(p_num));

    p_num++;
    printf("p_num ahora apunta a %d y su dirección es %d\n", *p_num, p_num);

    getchar();
    return (0);
}
```

Se pide:

- Descargar del AV el archivo denominado *pr5_ej8.c* al workspace Practico5. Crear un proyecto, compilarlo y ejecutarlo.
- Analizar gráficamente la salida del programa. ¿Los tamaños de los arreglos son iguales? ¿Qué diferencia en bytes existe entre las variables *p_num* y *p_let*?

Ej. 9– Dado el siguiente código:

```
# include <stdio.h>
int main (){
    struct medidas{
        float altura;
        int edad;
        float peso;
    };
    struct medidas reg1, reg2, *p_reg;
    printf("El tamaño de la estructura es %d y del puntero es %d \n",
        sizeof(reg1), sizeof(p_reg));

    getchar();
    return(0);
}
```

Se pide:

- Explicar la diferencia que existe entre **struct medidas**, **reg1** y ***p_reg**.
- Descargar del aula virtual el archivo denominado *pr5_ej9.c* a su workspace, compilarlo y ejecutarlo.
- Analizar gráficamente la salida del programa.
- Agregar a la struct medidas un campo denominado sexo, de tipo char.
- Compilar y ejecutar. Analizar si se modificó algo en la salida por pantalla. ¿Es lo que se esperaba?
- Agregar al código anterior antes de `getchar()`; las siguientes sentencias.

```
(*p_reg).edad = 10; // (3)
(*p_reg).altura = 1.415;
(*p_reg).peso = 48.35;
(*p_reg).sexo = 'M';

reg2 = reg1; // (1)
reg2.sexo = 'F';
reg2.peso = 34.3;
reg1.peso = reg2.peso; // (2)

printf(" %d\n", reg2.edad);
printf(" %f\n", reg2.altura);
printf(" %f\n", reg2.peso);
printf(" %f\n", reg1.peso);
```

- Agregar el código que sea necesario, compilar y ejecutar paso a paso (*depuración*).
- Responder si son correctas las sentencias indicadas en los comentarios (1) y (2). Explicar qué realiza cada una de ellas.
- Analizar los distintos valores que se van asignando a las variables.
- Escribir la sentencia señalada en (3) usando otra notación.

Ej. 10.- Codificar un programa que defina una estructura con los siguientes campos: *let_max*, *let_min*, de tipo carácter y *num* de tipo entero. **Posteriormente**, se deben declarar 2 variables del tipo definido e ingresar por teclado información para ambas variables. **Finalmente**, intercambiar los datos entre ambas variables y mostrar por pantalla sus contenidos modificados. Analizar si es posible intercambiar solo el campo *num* entre ambas variables. Codificar una función que implemente el intercambio de un campo entre dos variables de tipo estructura.

Nota: los datos del campo *let_min* se encuentran en el rango [a-z]. Los datos del campo *let_max* se encuentran en el rango [A-Z]. Los datos del campo *num* deben encontrarse en el rango [1-99].

Ej. 11.- Descargar del aula virtual el archivo *pr5_ej11.c* y realizar lo siguiente:

1. Crear un proyecto con el código fuente *pr5_ej11.c* y analizar su contenido. Compilar y ejecutar.
2. Colocar en el código fuente los siguientes comentarios donde corresponda:
 - a. Definición Global del tipo empleado.
 - b. Empleado es un tipo estructura con dos campos uno Código y el otro Nombre.
 - c. Uno de los parámetros actuales es "todos", un arreglo tipo estructura empleado.
 - d. Declaro "todos" como un arreglo tipo estructura empleado.
 - e. Uno de los parámetros formales es puntero del tipo struct empleado.
 - f. Acceso al campo "codigo" mediante puntero.
 - g. Acceso al campo "nombre" mediante puntero.
 - h. Imprime los datos ingresados.
 - i. Incremento el puntero.
3. Guardar los cambios.

Se pide responder:

- a) ¿De qué tipo son los campos de "empleado"?
- b) ¿Cuántas variables de tipo "empleado" se han declarado en el programa?
- c) ¿Puedo declarar más variables y/o arreglos de tipo "struct empleado"?
- d) ¿Cuántos datos de empleados se puede almacenar?
- e) ¿Por qué el primer parámetro formal de la función ingreso es un puntero?
- f) En la función ingreso, ¿qué formato de información permite ingresar en el campo nombre?
- g) Si `scanf("%d",&((*a).codigo));` es lo mismo que escribir `scanf("%d",&(a->codigo));`, ¿cuál es el equivalente para el campo nombre? Probar en el IDE.

Ej. 12- Realizar las siguientes modificaciones en el programa anterior:

- a) Agregar a la estructura empleado dos nuevos campos: Sueldo (float) y CategEmp (char) (Gerencia, Administración, Planta, Mantenimiento). **Nota:** *El sueldo no puede exceder los \$120000.*
- b) Modificar la función ingreso y la función main para poder ingresar información en los dos nuevos campos. **Realizar todos los controles necesarios.**
- c) Codificar una función que reciba una categoría de empleado y muestre por salida estándar los datos de los empleados de esa categoría. **Realizar todos los controles necesarios.**
- d) Dado un código ingresado por el usuario, codificar una función que muestre el nombre, la categoría (por ejemplo: Planta) y el sueldo de dicho empleado.
- e) Definir una función que permita aumentar el sueldo de un empleado un 10%, si dicho sueldo es menor que \$35000.
- f) Codificar un menú de opciones para que el usuario pueda realizar estas acciones (ítem e y f), las veces que desee. Dicho menú también debe contar con una opción para finalizar el programa.

Ej. 13- Modificar el proyecto del Ej. 13 del Práctico 2, para que cada ordenanza se almacene en una estructura que contenga los campos: *año, facultad, numOrd*. Codificar una función que permita buscar una ordenanza ingresada por el usuario e informar por salida estándar si fue ingresada o no. **Nota:** *para definir la estructura de datos a utilizar, recordar que se deben almacenar varias ordenanzas.*

Ej. 14- Modificar la estructura del ejercicio 9 agregando un campo denominado nombre y un campo denominado DNI, para que junto con los datos de edad, altura, sexo y peso, permita llevar el control físico de un estudiante. Codificar una función que permita informar los datos de aquellas estudiantes que sean de sexo femenino, su edad se encuentre entre 15 y 18 años y su altura sea menor a 150 cm.

Ej. 15- El gimnasio “GYM Power” brinda servicios deportivos/recreativos, en disciplinas tales como: natación, funcional, spinning, crossfit, halterofilia y cardio. El gerente del gimnasio solicita llevar un registro de su plantel de socias/os con los datos que correspondan. Codificar un programa que permita ingresar los datos necesarios para cada socio del gimnasio (*se puede usar la estructura modificada en el ejercicio anterior como base y agregar los campos que se considere necesario*). **Posteriormente**, el programa debe mostrar nombre, edad y disciplina de **los socios** mayores a 45 años, inscriptos en la disciplina halterofilia y **las socias** cuya altura sea mayor o igual a 170 cm, que practican natación. **Finalmente**, se debe mostrar por salida estándar un listado de todas las personas menores de edad (<18).

Ej. 16- Codificar un programa que permita almacenar una agenda de hasta 50 contactos. Los datos que se deben almacenar por cada contacto son: nombre y apellido (no más de 30 caracteres), correo electrónico (no más de 30 caracteres), teléfono móvil y fecha de nacimiento (dd/mm/aaaa). **Luego**, se deben separar los contactos almacenados en 2 arreglos diferentes: el primero contendrá nombre, apellido y número de teléfono móvil de los contactos cuyo año de nacimiento sea posterior al año 2000. El segundo arreglo contendrá la información completa de los contactos cuyo código de área sea de San Luis (266) y el año de nacimiento sea anterior al año 2000. **Finalmente**, mostrar la información de los tres arreglos con los carteles indicativos que corresponda.

Ejercicio Adicional

Ej. 1.– Realizar un programa que almacene en 3 arreglos de 10 posiciones (llamados A, B y C), datos cuyo código ASCII varíe en el rango de 70 a 90. **Luego**, el programa debe armar un único arreglo de estructuras llamado D3, donde cada posición del mismo contendrá los caracteres de los arreglos A, B y C en esa misma posición y en ese mismo orden. Finalmente se deberá mostrar por pantalla el contenido de los arreglos A, B, C y el arreglo resultante, con los carteles indicativos correspondientes.

**Licenciamiento:**

Un camino de encuentros: Prácticas Educativas Abiertas por RED ISEDU y Centro de Informática Educativa se distribuye bajo una Atribución-No Comercial Compartir Igual 4.0 Internacional.

**Usted es libre de:**

- Compartir - copiar y redistribuir el material en cualquier medio o formato • Adaptar remezcla, transformar y construir sobre el material En los siguientes términos:
 - Atribución: Usted debe reconocer los créditos de la obra de la manera especificada por el autor o el licenciante (pero no de una manera que sugiera que tiene su apoyo, o que apoyan el uso que hace de su obra).
 - No comercial - No puede utilizar esta obra para fines comerciales. • Compartir igual: La explotación autorizada incluye la creación de obras derivadas, siempre que mantengan la misma licencia al ser divulgadas.
 - No hay restricciones adicionales - No se pueden aplicar términos legales o medidas tecnológicas que restringen legalmente otros de hacer cualquier cosa que los permisos de licencia.
- Entendiendo que:
 - Renuncia — Alguna de estas condiciones puede no aplicarse si se obtiene el permiso del titular de los derechos de autor.
 - Dominio Público — Cuando la obra o alguno de sus elementos se halle en el dominio público según la ley vigente aplicable, esta situación no quedará afectada por la licencia.
 - Otros derechos — Los derechos siguientes no quedan afectados por la licencia de ninguna manera:
 - Los derechos derivados de usos legítimos u otras limitaciones reconocidas por ley no se ven afectados por lo anterior.- Los derechos morales del autor.
 - Derechos que pueden ostentar otras personas sobre la propia obra o su uso, como por ejemplo derechos de imagen o de privacidad.
 - Aviso — Al reutilizar o distribuir la obra, tiene que dejar muy en claro los términos de la licencia de esta obra. La mejor forma de hacerlo es enlazar a esta página.
<http://www.creativecommons.org.ar/licencias>