Anexo B. Ejercicios nivel 1 (sencillo-medio)

Introducción

En este anexo incluimos una colección de ejercicios que podríamos considerar de un nivel sencillo/medio. Eso no significa que sean fáciles. Si es tu primera toma de contacto con la materia, pueden requerir de mucho esfuerzo. Para solucionarlos, se deben conocer los tipos de datos simples y las estructuras de control, esto es, hasta el tema 3 incluido. Las soluciones de los ejercicios vienen al final del capítulo, pero no las consultes hasta que no los intentes solucionar por ti mismo.

Ejercicios

A continuación presentamos los enunciados de los ejercicios propuestos, ordenados de menor a mayor dificultad.

Ejercicio 1

Comenzaremos con el ejercicio más sencillo que existe en un lenguaje de programación, el "Hola Mundo". Consiste simplemente en realizar un programa completo en C que saque el mensaje por pantalla "Hola Mundo". Para enriquecerlo un poco, colocaremos el mensaje en el centro de la pantalla un poco adornado y esperaremos a que el usuario pulse una tecla.

Ejercicio 2

En este ejercicio vamos a solicitar un número flotante *n* al usuario y a continuación mostraremos varios cálculos con él, que serán: *-n*, *1/n* y $\sqrt{n}$. Ten en cuenta que la primera operación se puede realizar siempre, pero en los otros dos casos hay que tener en cuenta las excepciones, mostrando un mensaje de error adecuado si no se pueden realizar. Para calcular la raíz cuadrada de un número, utiliza la función *resul=sqrt(num);* de la librería *math.h*.

Ejercicio 3

En este ejercicio realizaremos un programa que calcule el Índice de Masa Corporal (IMC) de una persona, para estimar su perfil nutricional. Para ello, solicitaremos al usuario su peso, expresado en kilogramos y su altura expresada en metros. A continuación, imprimiremos su IMC, que se calcula como: **IMC=peso/(altura^2)**. En función de este valor imprimiremos su perfil, utilizando la siguiente tabla:

Valor de IMC	Perfil nutricional
Menor a 16	Deficiencia nutricional en tercer grado
Entre 16 y 17	Deficiencia nutricional en segundo grado
Entre 17 y 18.5	Deficiencia nutricional en primer grado
Entre 18.5 y 20	Bajo peso
Entre 20 y 25	Normal
Entre 25 y 30	Sobrepeso
Entre 30 y 35	Obesidad en primer grado
Entre 35 y 40	Obesidad en segundo grado
Mayor a 40	Obesidad en tercer grado

Ejercicio 4

Realizar un programa que solicite al usuario un número. A continuación nos informará:

a) Si es positivo, negativo o cero
b) Si es real o entero
c) En caso de ser entero, nos dirá el número de cifras que posee.
d) En caso de ser real mostrará por separado su parte entera y su parte fraccionaria.

Es necesario tener en cuenta que:

- En caso de ser cero, no se mostrarán los apartados b,c ni d.
- Toda la información anterior se mostrará en una única frase
- Se recomienda utilizar la función *sprintf(cadena, cadena de formato, parámetros)*, así como el *casting* (modificadores de tipo momentáneos).

Ejercicio 5

Realizar un programa completo en C que solicite 7 números decimales al usuario y muestre el mayor, el menor y la media de los mismos. Realizar el mismo ejercicio primero sin bucles y luego con bucles.

Ejercicio 6

Realizar un programa C que solicite al usuario un número y nos informe si es o no primo.

Ejercicio 7

Siguiendo la misma temática que en el ejercicio anterior, solicitar un número *n* al usuario y sacar por pantalla los *n* primeros números primos.

Ejercicio 8

Solicitar una cadena de caracteres al usuario y mostrar por pantalla: la cadena en mayúsculas, la cadena en minúsculas y la cadena sustituyendo todos los caracteres no alfabéticos por asteriscos.

Ejercicio 9

Solicitar una cadena de caracteres al usuario y mostrarla en el centro de la pantalla rotando sobre sí misma, al estilo de los carteles publicitarios. En cada iteración, se debe poner el primer carácter de la cadena en la última posición y desplazar el resto una unidad a la izquierda. A continuación, borramos la pantalla e imprimimos la cadena en el centro de la misma. El proceso se debe repetir indefinidamente (bucle infinito). Se aconseja introducir un pequeño retardo cada vez que se imprima la cadena con el fin de que el movimiento no sea excesivamente rápido (función *_sleep([milisegundos])*)

Ejercicio 10

Continuando con las animaciones, en este ejercicio realizaremos un programa que simule una pelota moviéndose en diagonal rebotando indefinidamente sobre los bordes de la pantalla. En cada iteración, borraremos la pantalla y colocaremos la pelota en su posición (la pelota será, por ejemplo, la letra 'o'). A continuación, se actualizará la posición teniendo en cuenta que al llegar a cualquier borde de la pantalla hay que cambiar la dirección de la misma, consiguiendo el efecto de rebote. Se aconseja utilizar dos variables para mantener la posición de la pelota (inicializadas al centro de la pantalla al principio) y otras dos variables para mantener el vector con la dirección de la misma (inicializadas para que la primera dirección sea diagonal hacia abajo). De esta forma, para mover la pelota se debe sumar a la posición el vector dirección. El proceso se debe repetir indefinidamente. De nuevo, se aconseja utilizar la función *_sleep([milisegundos])* para introducir un pequeño retardo, de forma que la animación no sea demasiado rápida.

Ejercicio 11

Solicitar un número entero al usuario y mostrar por pantalla su descomposición en factores primos.

Soluciones

A continuación presentamos las soluciones a los ejercicios propuestos. Recuerda que normalmente hay varias soluciones al mismo ejercicio. Aquí solo presentamos una solución válida, que no ha de ser la mejor. No consultes la solución hasta que no agotes las posibilidades, porque aprender a programar no consiste en comprender la solución sino en llegar a ella por ti mismo. Utiliza el compilador para probar tus soluciones.

Solución ejercicio 1

Una posible solución sería:

```
//librerías necesarias
#include <stdio.h>  //para utilizar printf
#include <conio.h>  //para utilizar getche

//función principal
int main(void){
   //con intros centramos en vertical
   printf("\n\n\n\n\n\n\n\n\n\n");
   //con tabuladores y espacios centramos en horizontal
   printf("\t\t\t    ----------------------\n");
   printf("\t\t\t         Hola Mundo\n");
   printf("\t\t\t    ----------------------\n");

   //esperamos la pulsación de una tecla
   getche();
   //finalizamos
   return(0);
}
```

Solución ejercicio 2

Una posible solución sería:

```
//librerías necesarias
#include <stdio.h>  //para printf y scanf
#include <conio.h>  //para utilizar getche
#include <math.h>   //para utilizar sqrt

//función principal
int main(void){
   //variables
   float n;
   //solicitamos el número al usuario
   printf("Introduzca numero: ");
   scanf("%f",&n);
```

```
    //-n se puede hacer directamente
    printf("El resultado de -n = %f\n",-n);
    //1/n solo si es distinto de cero
    if(n!=0){
       printf("El resultado de 1/n = %f\n",1/n);
    }else{
       printf("ERROR: 1/n no se puede realizar porque n es cero\n");
    }
    //raiz(n) solo si es positivo
    if(n>=0){
       //sqrt(num) devuelve la raíz cuadrada de número (librería math.h)
       printf("El resultado de raiz(n) = %f\n", sqrt(n) );
    }else{
       printf("ERROR: sqrt(n) no se puede realizar porque n es negativo\n");
    }

    //esperamos la pulsación de una tecla
    getche();
    //finalizamos
    return(0);
}
```

Solución ejercicio 3

Una posible solución consiste en calcular el IMC y clasificarlo a través de una secuencia de if-else anidados:

```
//librerías necesarias
#include <stdio.h>  //para utilizar printf
#include <conio.h>  //para utilizar getche

//función principal
int main(void){
   //pedimos el peso y la altura
   float peso, altura, imc;
   printf("Introduzca su peso en kilogramos: ");
   scanf("%f",&peso);
   printf("Introduzca su altura en metros: ");
   scanf("%f",&altura);

   //imc
   if(altura>0){
      imc=peso/(altura*altura);
      printf("Indice de masa corporal: %2.2f\n",imc);
      printf("Perfil nutricional: ");
      if(imc<16) printf("Deficiencia nutricional en tercer grado");
      else if(imc<17) printf("Deficiencia nutricional en segundo grado");
      else if(imc<18.5) printf("Deficiencia nutricional en primer grado");
      else if(imc<20) printf("Bajo peso");
      else if(imc<25) printf("Normal");
      else if(imc<30) printf("Sobrepeso");
```

```
        else if(imc<35) printf("Obesidad en primer grado");
        else if(imc<40) printf("Obesidad en segundo grado");
        else printf("Obesidad en tercer grado");
        printf("\n");
    }else{
        printf("Error: altura incorrecta\n");
    }

    //esperamos la pulsación de una tecla
    getche();
    //finalizamos
    return(0);
}
```

Solución ejercicio 4

Una posible solución sería la siguiente:

```
//librerías necesarias
#include <stdio.h>  //para utilizar printf
#include <string.h> //para funciones de cadenas
#include <conio.h>  //para utilizar getche

//función principal
int main(void){
    //variables
    float num;
    int nc;

    //solicitamos el número
    printf("Introduzca un numero: ");
    scanf("%f",&num);

    //vemos si es cero
    if(num==0){
        printf("Se trata del numero cero\n");
    }else{
        //positivo o negativo
        printf("Se trata de un numero ");
        if(num>0){
            printf("positivo, ");
        }else{
            printf("negativo, ");
        }
        //vemos si es real o entero
        if(num==(float)((int)num)){
            printf("entero ");
            //numero de cifras
            char aux[100];
            sprintf(aux,"%d",(int)num);
            nc=strlen(aux);
```

```
            //en caso de negativo descontamos
            //el signo -
            if(num<0) nc=nc-1;
            printf("y compuesto por %d cifras.\n",nc);
        }else{
            printf("real, ");
            //parte real y fraccionaria
            printf("con parte real %d y fraccionaria %f.\n",(int)num,num-(int)num);
        }
    }

    //esperamos la pulsación de una tecla
    getche();
    //finalizamos
    return(0);
}
```

Solución ejercicio 5

Una posible solución sin bucles sería la siguiente:

```
//librerías necesarias
#include <stdio.h>  //para utilizar printf
#include <conio.h>  //para utilizar getche

//función principal
int main(void){
    //variables
    float n1,n2,n3,n4,n5,n6,n7,mayor,menor,media;
    //solicitamos los 7 números
    printf("Introduzca numero 1 :");
    scanf("%f",&n1);
    printf("Introduzca numero 2 :");
    scanf("%f",&n2);
    printf("Introduzca numero 3 :");
    scanf("%f",&n3);
    printf("Introduzca numero 4 :");
    scanf("%f",&n4);
    printf("Introduzca numero 5 :");
    scanf("%f",&n5);
    printf("Introduzca numero 6 :");
    scanf("%f",&n6);
    printf("Introduzca numero 7 :");
    scanf("%f",&n7);


    //calculamos el mayor
    mayor=n1;
    if(n2>mayor) mayor=n2;
    if(n3>mayor) mayor=n3;
    if(n4>mayor) mayor=n4;
```

```
    if(n5>mayor) mayor=n5;
    if(n6>mayor) mayor=n6;
    if(n7>mayor) mayor=n7;
    printf("El mayor es %f\n",mayor);

    //calculamos el menor
    menor=n1;
    if(n2<menor) menor=n2;
    if(n3<menor) menor=n3;
    if(n4<menor) menor=n4;
    if(n5<menor) menor=n5;
    if(n6<menor) menor=n6;
    if(n7<menor) menor=n7;
    printf("El menor es %f\n",menor);

    //calculamos la media
    media=(n1+n2+n3+n4+n5+n6+n7)/7;
    printf("La media es %f\n",media);

    //esperamos la pulsación de una tecla
    getche();
    //finalizamos
    return(0);
}
```

Como vemos, la solución sin bucles nos obliga a crear muchas variables. Al poder utilizar bucles, el programa resulta más compacto y extensible a más números:

```
//librerías necesarias
#include <stdio.h>  //para utilizar printf
#include <conio.h>  //para utilizar getche

//definiciones
#define N 7

//función principal
int main(void){
   //variables
   int i;
   float num,mayor,menor,media=0;
   //solicitamos los números y a la vez
   //calculamos maximo minimo y media
   for(i=0;i<N;i++){
      printf("Introduzca numero %d :",i+1);
      scanf("%f",&num);
      if(i==0){
         //primera iteración
         mayor=menor=media=num;
      }else{
         //resto de iteraciones
         if(num>mayor) mayor=num;
```

```
            if(num<menor) menor=num;
            media=media+num;
        }
    }

    //resultados
    printf("El mayor es %f\n",mayor);
    printf("El menor es %f\n",menor);
    printf("La media es %f\n",media/N);

    //esperamos la pulsación de una tecla
    getche();
    //finalizamos
    return(0);
}
```

Solución ejercicio 6

A continuación presentamos una posible solución, que consiste en comprobar si el número introducido *n* es divisible por algún número entre *2* y *n-1*. Para ello, utilizamos un bucle con un centinela. Notar la doble condición en el bucle, que hace que se corte cuando encuentra el primer divisor válido.

```
//librerías necesarias
#include <stdio.h>  //para utilizar printf
#include <conio.h>  //para utilizar getche
//función principal
int main(void){
    //variables
    int num,i,centinela;
    //pedimos el número
    printf("Introduzca numero: ");
    scanf("%d",&num);

    //recorremos del 2 al num-1 para ver si
    //es divisible por alguno
    centinela=0;
    for(i=2;i<num && centinela==0;i++){
        if(num%i==0){
            centinela=1;
        }
    }

    if(centinela==1){
        printf("El numero no es primo\n");
    }else{
        printf("El numero es primo\n");
    }
     //esperamos un intro y finalizamos
    getche();
```

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

Solución ejercicio 7

En este caso, utilizamos un bucle con condición inicial que lleva la cuenta de cuántos números primos hemos impreso por el momento, hasta completar los que ha solicitado el usuario:

```
//librerías necesarias
#include <stdio.h>  //para utilizar printf
#include <conio.h>  //para utilizar getche
//función principal
int main(void){
   //variables
   int num,i,centinela,actual,nprimos;
   //pedimos el número
   printf("Introduzca numero: ");
   scanf("%d",&num);
   //inicializamos el número de primos y el
   //número actual
   nprimos=0;
   actual=1;
   printf("Los %d primeros numeros primos son: ",num);
   while(nprimos<num){
      //comprobamos si actual es primo
      centinela=0;
      for(i=2;i<actual && centinela==0;i++){
         if(actual%i==0){
            centinela=1;
         }
      }
      //si es primo lo mostramos y aumentamos nprimos
      if(centinela==0){
         printf("%d ",actual);
         nprimos++;
      }
      actual=actual+1;
   }
   //esperamos un intro y finalizamos
   getche();
   return(0);
}
```

Solución ejercicio 8

El ejercicio se puede realizar con o sin una cadena adicional. En esta solución nos decantamos por usar una cadena auxiliar en la que hacemos los cambios que se realizan sobre la cadena original. Te invitamos a que hagas el mismo ejercicio con una sola cadena.

```
//librerías necesarias
```

```
#include <stdio.h>  //para utilizar printf
#include <conio.h>  //para utilizar getche
#include <string.h> //para utilizar cadenas
//función principal
int main(void){
   //variables
   int i,l;
   char cad[100],aux[100];
   //pedimos la cadena
   printf("Introduzca texto: ");
   //con gets pedimos una cadena que puede contener espacios.
   //Para cadenas es mejor que scanf
   gets(cad);
   //medimos la longitud
   l=strlen(cad);
   //colocamos el fin de cadena de aux en el mismo sitio que la original
   aux[l]='\0';

   //copiamos de cad a aux pasando las minúsculas a mayúsculas
   for(i=0;i<l;i++){
      if(cad[i]>='a' && cad[i]<='z'){
         aux[i]=cad[i]-'a'+'A';
      }else{
         //el resto queda igual
         aux[i]=cad[i];
      }
   }
   //Imprimimos el resultado
   printf("Cadena en mayusculas : %s\n",aux);

   //el mismo proceso de mayúsculas a minúsculas
   for(i=0;i<l;i++){
      if(cad[i]>='A' && cad[i]<='Z'){
         aux[i]=cad[i]-'A'+'a';
      }else{
         //el resto queda igual
         aux[i]=cad[i];
      }
   }
   //Imprimimos el resultado
   printf("Cadena en minusculas : %s\n",aux);

   //copiamos sustituyendo no alfabéticos por *
   for(i=0;i<l;i++){
      //si es mayúscula o minúscula lo copiamos igual
      if( (cad[i]>='a' && cad[i]<='z') || (cad[i]>='A' && cad[i]<='Z')){
         aux[i]=cad[i];
      }else{
         //el resto cambia por '*'
         aux[i]='*';
```

```
            }
        }
        //Imprimimos el resultado
        printf("Cadena sin caracteres alfabeticos : %s\n",aux);

        //esperamos un intro y finalizamos
        getche();
        return(0);
    }
```

Solución ejercicio 9

De nuevo, este ejercicio se puede realizar con dos cadenas de caracteres, pero en este caso hemos optado por una solución con una única cadena:

```
//librerías necesarias
#include <stdio.h>  //para utilizar printf
#include <string.h> //para utilizar cadenas
#include <stdlib.h> //para utilizar system
//función principal
int main(void){
    //variables
    int i,l;
    char cad[100];
    //pedimos la cadena (con gets mejor que con scanf)
    printf("Introduzca texto: ");
    gets(cad);
    //medimos su longitud
    l=strlen(cad);

    //bucle infinito
    while(1){
        //desplazamos la frase un caracter
        char aux=cad[0]; //primer carácter
        for(i=0;i<l-1;i++) cad[i]=cad[i+1]; //resto
        cad[l-1]=aux; //colocamos el primero al final

        //borramos la pantalla y nos posicionamos en el centro
        system("cls");

        //bajamos 12 filas (la pantalla tiene 24)
        for(i=0;i<12;i++) printf("\n");
        //para centrar el texto colocamos 40 espacios menos
        //la mitad de la longitud (la pantalla tiene 80 columnas)
        for(i=0;i<40-l/2;i++) printf(" ");

        //imprimimos la cadena
        printf("%s",cad);

        //introducimos un pequeño retardo de 100 milisegundos
```

```
        _sleep(100);
    }
    return(0);
}
```

Solución ejercicio 10

Una posible solución sería la siguiente:

```
//librerías necesarias
#include <stdio.h>  //para utilizar printf
#include <string.h> //para utilizar cadenas
#include <stdlib.h> //para utilizar system

//función principal
int main(void){
    //variables
    int i,pi,pj,vi,vj;
    //posición
    pi=20;
    pj=40;
    //dirección
    vi=vj=1;

    //bucle infinito
    while(1){
        //borramos la pantalla
        system("cls");
        //nos posicionamos en (pi,pj)
        for(i=0;i<pi;i++) printf("\n");
        for(i=0;i<pj;i++) printf(" ");
        //imprimimos la pelota
        printf("o");
        //pequeño retardo de 50 milisegundos
        _sleep(50);

        //actualizamos la dirección
        if(pi>=24 || pi<=0) vi*=-1;
        if(pj>=80 || pj<=0) vj*=-1;

        //actualizamos la posición
        pi=pi+vi;
        pj=pj+vj;
    }
    return(0);
}
```

Solución ejercicio 11

Para solucionar este ejercicio, generamos una secuencia de números creciente (variable *divisor*) con la que probamos a dividir a nuestro número. Si el número es divisible por el candidato, lo dividimos y mostramos el resultado de la división. Si no es divisible,

incrementamos *divisor* para ir probando con otros números. Conforme vamos dividiendo al número, va disminuyendo. El bucle principal termina cuando el número se hace 1. Notar que, aunque no se ha programado de forma explícita, los únicos divisores que pueden aparecer por pantalla son primos. Este ejercicio se podría optimizar mucho, intentando disminuir el considerable número de divisiones que se realiza.

```
//librerías necesarias
#include <stdio.h>  //para utilizar printf
#include <conio.h>  //para utilizar getche

//función principal
int main(void){
   //variables
   int n,divisor;

   //solicitamos el número
   printf("Introduzca numero entero: ");
   scanf("%d",&n);

   printf("Descomposicion en factores primos de %d = ",n);

   //caso especial cuando n==1
   if(n==1) printf("1\n");

   //divisor con el que vamos probando
   divisor=2;
   //bucle principal
   while(n>1){
      if(n%divisor==0){
         //si es divisible por divisor, dividimos n,
         //imprimimos el resultado y no modificamos divisor
         n=n/divisor;
         //dependiendo si es el último o no imprimimos la X
         if(n>1) printf("%d X ",divisor);
         else    printf("%d ",divisor);
      }else{
         //si no es divisible debemos probar con otro
         divisor++;
      }
   }
   getche();
   return(0);
}
```