Billy Yong Le López Lin 1CV6

2da lista de problemas de Fundamentos de programación (abstracta).

0. Codifique, utilizando operaciones de aritmética de punto flotante, las siguientes fórmulas físicas e indique los valores máximos y mínimos que puede representar cada variable.

   a. $\vec{F} = m\omega^2\vec{r}$
   b. $\vec{F} = m\vec{g} - k\vec{x}$
   c. $\vec{F} = q\vec{E} + q\vec{v} \times \vec{B}$
   d. $\theta = \theta_0 + \omega t + \frac{1}{2}\omega t^2$

   El tipo de dato de punto flotante tiene un tamaño de 32 bits o 4 bytes, y si consideramos que no se utilizarán números negativos, su rango de valores es de 0.0 hasta 2^{32} o 4, 294, 967, 295; si consideramos a los valores negativos, entonces el valor mínimo sería -2, 147, 483, 648 hasta 2, 147, 483, 647. Para algunas fórmulas se utilizará el rango positivo y en otras el rango con negativos.

   a. Esta fórmula pertenece a la del movimiento circular. Iniciaré con los valores de F; debido a que este será el resultado de la operación, su valor no debe de superar 2^{32}, y como en la práctica esta variable debería ser positivo, entonces su valor máximo es 2^{32} y el mínimo sería 0.0. Además, por tratarse de un vector, tendrá sus tres componentes, con cada componente con su rango de valores 2^{32}.

      Esto nos lleva a decir que las demás variables deberán ser menores o igual a 2^{32}. Intentando ser lo más equitativo para las variables, podrían tener un rango de valores de 2^8 cada una; 2^8 para la masa m, 2^{16} para ω^2 (porque es el producto de ω con sí misma), 2^8 para r, el cual es un vector de posición y tiene sus tres componentes con rango de valores 2^8. El rango mínimo para todas las variables sería 0.

   b. La fórmula es de la ley de elasticidad de Hooke. Tenemos las variables m, masa del objeto; g, vector aceleración de la gravedad, la cual es negativa (por estar en vertical); k, el coeficiente de resorte; x, vector posición con tres componentes; y, por supuesto, F, la fuerza con sus tres componentes.

      Como la masa no puede ser negativa, su producto con la aceleración es negativo, y la suma de éste con –kx es, probablemente, negativo, entonces el rango de valores de F como mínimo tendríamos -2^{31} y como máximo sería $2^{31} – 1$ (ya que alguna componente de x nos puede resultar negativa y provocaría que –kx se vuelva positiva).

      Ahora, el rango de las variables podría suponerse de esta manera: la masa, que debe ser positiva, tendría como valor mínimo valores muy cercanos a 0.0, y como máximo 109, 565, 492.2, esto es porque si lo multiplicamos por -9.8 (la aceleración) nos dará -

2^{30} y debido a que se sumará, entonces es posible que el resultado sea el mínimo o el máximo de F; el vector aceleración es constante con -9.8 en su componente en z, pero en x e y son 0.0; k puede variar mucho, pero siempre es un valor positivo, por lo que su valor mínimo puede ser cercano a 0.0 y su máximo sería 2^{15} y esto se sabrá después; con el vector de posición x tenemos que su valor mínimo puede ser -2^{15} y como máximo 2^{15}, escogí estos valores ya que el producto del valor máximo de k y el de x sería 2^{30}, y si mg llegase a ser 2^{30}, entonces la suma entre estos dos sería 2^{31}, el máximo de F.

c. Esta es la fórmula de la fuerza de Lorentz. Tenemos que q es la carga eléctrica con valor de 1.60217662×10^{-19}; E es el campo eléctrico, el cual es un vector con tres componentes; v es la velocidad, que también es un vector de tres componentes; B es el campo magnético, que igualmente es un vector. La fuerza F puede tener valores negativos, entonces su mínimo sería -2^{31} y su máximo $2^{31} - 1$.

   Tomando de referencia la fórmula b, cada componente de E tendrá un rango mínimo de $-3.350884698\times10^{27}$ y máximo de 3.350884697×10^{27}, estos valores se obtuvieron al dividir 2^{30} entre q, y estos al poder tener valores negativos, entonces ese resultado se divide entre 2 para obtener el mínimo, y el máximo sería el mismo resultado pero -1. Y puesto a que la fuerza eléctrica se sumará con la fuerza magnética, entonces es posible que la fuerza magnética adquiera el valor de 2^{30}, así la suma sería el valor máximo de F.

   Por otra parte, para la fuerza magnética, podemos suponer que el mínimo de cada variable es 1.675442349×10^{27}, y el máximo sería 1.675442348×10^{27}. Este resultado se obtuvo dividiendo a la mitad el rango de valores de E, ya que estos valores se repartirán entre v y B, y así se intenta ser lo más equitativo posible. Así, cada componente de v y B tendrá su rango de valores ya mencionado.

d. La fórmula corresponde al desplazamiento angular. θ es el desplazamiento angular; θ_0 es el desplazamiento inicial; ω, la velocidad angular; t, el tiempo. Digamos que θ puede ser negativo porque su desplazamiento puede darse en un sentido negativo, siendo esto así, entonces su mínimo es -2^{31} y el máximo $2^{31} - 1$.
   Por otra parte, θ_0 igualmente puede optar por valores negativos, entonces su mínimo sería -357,913,941.2 y el máximo 357,913,940.2; esto se obtuvo considerando que la suma de los máximos dé $2^{31} - 1$, por lo que opté por dividir esa cantidad entre 3, y por tener la posibilidad de ser negativo, fue dividido por la mitad.

   Para la velocidad angular, éste podría obtener valores negativos debido a un desplazamiento por el sentido opuesto, y consideré que podría darse la posibilidad de que tanto ω como t sean casi iguales, por lo que decidí tomar raíz cuadrada de θ_0, así su producto sería lo más similar a θ_0. Sin embargo, como la velocidad puede ser

negativa, entonces lo dividí por la mitad, así que el rango de valores para la velocidad angular sería como mínimo -9,459.306808 y máximo 9,458.306808; para el tiempo, éste no puede ser negativo, así que su mínimo sería un valor cercano a 0.0 y máximo 18,918.61362.

Para la última parte de la fórmula, los rangos de las variables ya están dados, así que solo queda mencionar que el producto no superará el límite del rango máximo pues se dividirá entre dos.

1. Escriba el valor de z para los siguientes casos: a) x = 0, y = 0, b) x = 0, y = 1, c) x = 1, y = 0 y d) x = 1, y = 1

a.
```
if (((x==0) && (y==1)) || ((x==1) && (y==0)))
  z= 1;
else
  z= 0;
```

b.
```
if (((x==1) || (y==0)) && ((x==0) && (y==1)))
  z= 1;
else
  z= 0;
```

c.
```
if (x==0)
  {
      if (y==0)
        z= 0;
      else
        z= 1;
  }
else
  {
      if (y==0)
        z= 2;
      else
        z= 3;
  }
```

a) Teniendo x=0, y=0.
   a. Siendo la condición “x= 0 y y= 1” o “x= 1 y y= 0”, entonces no se cumple la condición puesto que el AND en ninguno de ambos es verdadero, por lo tanto, z=0.
   b. “x=1 o y=0” es verdadero, pero “x=0 y y=1” es falso, por lo que toda la conjunción es falsa y z=0.
   c. Como x=0, la primera condición se cumple; además, se cumple que y=0, así que z=0.

b) Teniendo x=0, y=1.

a. La primera condición “x= 0 y y= 1” es verdadera, y por ser un OR, toda la condición es verdadera, incluso si la segunda condición realmente es falsa. Esto es, z= 1.
b. La condición “x=1 o y=0” y “x=0 y y=1” es falso, debido a que “x=1 o y=0” es falso, por el AND esto hace que todo sea falso, incluso si “x=0 y y=1” es verdadero. Por tanto, z= 0.
c. Puesto que x=0, la primera condición se cumple, pero y=1, lo que no cumple la siguiente condición; entonces, z=1.

c) Sabiendo que x=1, y=0.
a. La primera condición no se cumple, pero la segunda sí, y por tener un OR de por medio, toda la condición se cumple. Por ende, z= 1.
b. La condición “x=1 o y=0” se cumple, pero “x=0 y y=1” es falsa, por consecuencia, todo es falso; esto implica z=0.
c. Inicialmente, como x=1, la primera condición no se cumple, lo que nos lleva a la condición si y=0, cosa que es verdadero; por lo tanto, z= 2.

d) Con x=1, y=1.
a. Puesto que ambas condiciones tienen un AND, ambas son falsas y, por ende, todo es falso y obtenemos que z=0.
b. La primera condición sí se cumple por el OR, pero la segunda no se cumple, y como hay un AND de por medio y solo una es verdadera, entonces todo es falso y z=0.
c. La primera condición de x=0 no se cumple, y la segunda condición de y=0 tampoco se cumple, esto nos lleva a z= 3.

2. Indique valor que adquiere la variable c dentro y fuera de los siguientes ciclos para N = 0, 1, 10, 30, 60, 90, 100. Codifíquelos nuevamente en términos de ciclos for.

```
a.
int c= 0;
while (c<N)
{
  c+=10;
}
printf(“%d”, c);
```

```
b.
int c= 0;
while (1)
{
  if (c==N)
    break;
  c+=10;
}
printf(“%d”, c);
```

```
c.
int c= 0;
while (1)
{
  c+=10;
  if (c==N)
    break;
}
printf(“%d”, c);
```

a. Para N=0, c no es menor a N, por lo que no pasa por el while e imprime “0”.

Para N=1, c es menor a N, la suma da c=10, entonces c es mayor a N, por lo que abandona el bucle e imprime “10”.

Para N=10, c es menor a N, así que procede a la suma; c adquiere el valor de 10, pero ya no se cumple la condición, por lo que imprime “10”.

Para N=30, c es menor a 30, así que la suma da c=10, y como se sigue cumpliendo la condición, entonces c=20; debido a que c sigue siendo menor a N, procede a la suma nuevamente. Cuando c=30, ya no se cumple la condición e imprime “30”.

Para N=60, guiándose por los anteriores, se puede intuir que se imprime “60”, ya que cuando c valga 60, el ciclo se termina.

Para N=90, cuando c obtiene el valor de 90, se deja de cumplir la condición y se imprime “90”

Para N=100, cuando c es 100, se imprime “100”.

En su equivalencia en “for” sería:

```
int c=0, N=0, i=0;

for (i=0; c<N; i++)

        c+=10;

printf (“%d”, c);
```

b. Para N=0, la condición evalúa si c es igual a N, es decir, c es igual a 0, lo cual es verdadero, por lo que el ciclo se rompe e imprime “0”.

Para N=1, la condición evalúa si c es igual a 1, lo cual es falso y ejecuta la suma, pero ahora c=10 y se sigue sin cumplir la condición de c igual a 1, por lo que sigue sumando infinitamente al no poder cumplirse la condición; esto hace que se vuelva un bucle infinito y nunca imprima c.

Para N=10, como no se cumple la condición “c igual a 10” entonces procede a realizarse la suma. Ya con c=10, la condición sí se cumple y se rompe el ciclo para terminar imprimiendo “10”.

Para N=30, puesto que c no es igual a 30, entonces la suma se realiza hasta que así sea; con c=30, la condición rompe el ciclo y se imprime “30”.

Para N=60, la condición no se cumple sino hasta que c sea igual a 60, por lo que se repite la suma hasta llegar a c=60, en donde se termina el ciclo y se imprime “60”.

Para N=90, la suma se realiza hasta 9 veces debido a que no se cumplía la condición “c igual a 90”; ya con c=90, ocurre el break del ciclo e imprime “90”.

Para N=100, c va sumando 10 cada que no se cumple la condición, y como c debe ser igual a 100, se repite la suma hasta 10 veces. Una vez siendo c=100, el ciclo acaba e imprime “100”.

En su equivalencia “for” sería:

```
int c=0, i=0, N=0;
for (i=0; i > 0; i++)
{
        if (c == N)
                break;
        c +=10;
}
Prinft (“%d”, c);
```

c. Para N=0, al inicio del ciclo, se realiza la suma de c y 10, haciendo que c=10; debido a esto, la condición no se cumple, pues 10 no es igual a 0, por lo que el ciclo se vuelve infinito al estar repetidas veces sumando 10 y nunca imprimiría el valor de c.

Para N=1, debido a que c se suma con 10, la condición de c es igual a 1 no se cumple, pues 10 no es igual a 1; esto provoca que el ciclo siga sumando sin llegar a terminar y, por consecuencia, sin llegar a imprimir c.

Para N=10, la suma de c y 10 da c=10, y esto cumple la condición de c igual a 10, así que el ciclo termina y se imprime “10”.

Para N=30, al principio, c=10 y esto no cumple la condición, por lo que se vuelve al inicio y se suma 10, teniendo c=20; pero esto no cumple la condición, así que se vuelve a sumar, teniendo c=30, siendo esto lo suficiente para que se cumpla la condición y se detenga el ciclo, imprimiendo “30”.

Para N=60, c inicia con 10, pero esto no cumple la condición, así que la suma se repite 5 veces más hasta llegar a c=60. Con esto la condición se cumple y el ciclo acaba, después imprime “60”.

Para N=90, la condición no se cumple al principio, así que la suma se repite 8 veces más hasta que c tenga el valor de 90. Entonces, la condición se cumple, se rompe el ciclo e imprime “90”.

Para N=100, la suma se hace hasta 10 veces para que “c igual a 100” se cumpla; el ciclo termina e imprime “100”.

En su equivalencia “for” sería:

```
int c=0, i=0, N=0;
for (i=0; i > 0; i++)
{
c +=10;
if (c == N)
        break;
}
Printf (“%d”, c);
```

3. ¿Qué valores va adquiriendo la variable k cuando N = 0, 1, 10, 30, 60, 90, 100? Codifíquelos nuevamente en términos de ciclos while.

a.
`int k=0;`
`for (k=0; k<N; k++)`
`  printf(“%d\n”, k);`
`printf(“final= %d\n”, k);`
	b.
	`int k=0;`
	`for (k=N; k>=0; k--)`
	`  printf(“%d\n”, k);`
	`printf(“final= %d\n”, k);`

c.
`int k=1;`
`for (k=1; k<N; k*=2)`
`  printf(“%d\n”, k);`
`printf(“final= %d”, k);`
	d.
	`int k=0;`
	`for (k=N; k>0; k/2)`
	`  printf(“%d\n”, k);`
	`printf(“final= %d”, k);`

a. Para N=0, k inicia con 0, entonces la condición k<0 no se cumple y no se realiza el bucle. Termina imprimiendo “final= 0”.

   Para N=1, como k es menor a 1, entonces imprime “0”, luego k se vuelve 1, así que no se cumple la condición y acaba el bucle. Imprime “final= 1”.

   Para N=10, con k siendo menor a 10, se imprimen los números del 0 hasta 9 conforme k vaya aumentando; pero cuando k sea 10, la condición ya no se cumplirá y finalizará el bucle. Se imprime “final= 10”.

   Para N=30, se imprimen los números del 0 al 29; cuando k=30, la condición deja de cumplirse y pasa a imprimir “final= 30”.

   Para N=60, el bucle acaba cuando k=60, por lo que antes de que la suma llegue a ese valor, se logran imprimir los números del 0 al 59; una vez terminado el bucle, se imprime “final= 60”.

Para N=90, son impresos los números del 0 al 89; con k=90 se finaliza el bucle y se imprime “final= 90”.

Para N=100, mientras se van haciendo las sumas de k, se van imprimiendo los números empezando desde 0; cuando se imprime el 99, el contador de k llega a 100 y acaba el bucle y es imprimido “final= 100”.

Su equivalente en “while”:

```
int k=0, N=0;
while (k<N)
{
Printf (“%d\n”, k);
K++;
}
Printf (“final= %d\n”, k);
```

b. Para N=0, dado que k=N, 0 en este caso, y la condición es que deba ser mayor o igual a 0, entonces se imprime “0”; luego se resta 1, lo que hace que k=-1 y la condición ya no se cumpla. Entonces, se acaba el bucle y se imprime “final= -1”.

   Para N=1, se imprime “1” y “0”, acaba el bucle porque k=-1 y se imprime “final=-1”.

   Para N=10, se imprime de forma regresiva los números del 10 al 0 y cuando k=-1 se imprime “final= -1”.

   Para N=30, son impresos los números del 30 al 0 hasta que k=-1; cuando se llegue a ese valor se imprimirá “final= -1”.

   Para N=60, se despliegan los números del 60 hasta el 0, y cuando k=-1, es desplegado “final= -1”.

   Para N=90, se imprime el conteo de 90 hasta 0, luego se imprime “final=-1” cuando K=-1.

   Para N=100, imprime los números del 100 al 0, y cuando se imprima “final=-1” k tendrá valor de -1.

   Su equivalencia en “while” es:

```
Int N=0, k=N;
While (k>=0)
{
```

```
        Printf (“%d\n”, k);
        k--;
    }
    Printf (“final= %d\n”, k);
```

c. Para N=0, la condición es que k sea menor a N, pero como k=1, entonces la condición no se cumple y pasa a imprimir “final= 1”.

   Para N=1, k y N son iguales, pero k no es menor que N, así que no se cumple la condición y pasa a imprimir “final= 1”.

   Para N=10, k sí es menor a 10, por lo que se imprime “1”, luego se realiza la multiplicación, esto da k=2; como k sigue siendo menor a 10, entonces sigue imprimiendo el valor de k, “2”; de esta forma va imprimiendo en potencias de 2, “4”, “8”. Cuando k=16, se deja de cumplir la condición y se imprime “final= 16”.

   Para N=30, la condición se cumple, así que se imprimen “1, 2, 4, 8, 16”, con k=32 ya no es válido. Se imprime “final= 32”.

   Para N=60, la condición se cumple, por lo que se imprime “1, 2, 4, 8, 16, 32”; en k=64 ya no se cumple la condición. Se imprime “final= 64”.

   Para N=90, la condición se cumple, por tanto, se imprime “1, 2, 4, 8, 16, 32, 64”; 128 es mayor que 90, así que se termina el bucle y se imprime “final= 128”.

   Para N=100, la condición se cumple, por tanto, se imprime “1, 2, 4, 8, 16, 32, 64”; como 128 es mayor que 100, entonces acaba el bucle y se imprime “final= 128”.

   Su equivalencia en “while” es:

```
    Int k=1, N=0;
    While (k<N)
    {
        Printf (“%d\n”, k);
        K*=2;
    }
    Printf (“final= %d\n”, k);
```

d. Para N=0, k adquiere el valor de N, y la condición es que k esa mayor a 0, pero esto no se cumple, por tanto, el bucle acaba y se imprime “final= 0”.

Para N=1, k es 1 y es mayor a 0, por tanto, se imprime “1”, luego se realiza la división entre 2, pero esto da 0.5 y como se está trabando con enteros, entonces sería como si k fuera 0, por lo que se acaba el bucle y se imprime “final= 0”.

Para N=10, k es 10 y es mayor a 0, por tanto, se imprime “10”, la división de k entre 2 es “5”, su mitad es 2.5, pero solo se considera la parte entera, así que es “2”, luego obtenemos “1”, y como la forma entera de 0.5 es 0, se termina el bucle y se imprime “final= 0”.

Para N=30, se imprime “30, 15, 7, 3, 1” y el bucle acaba cuando la división da 0.5, por lo que imprime “final= 0”.

Para N=60, se imprime “60, 30, 15, 7, 3, 1” y tras acabar el bucle se imprime “final= 0”.

Para N=90, es imprimido “90, 45, 22, 11, 5, 2, 1” y al terminar el bucle se imprime “final= 0”.

Para N=100, se imprime “100, 50, 25, 12, 6, 3, 1” y cuando termina el bucle se imprime “final= 0”.

Su equivalencia en “while” es:

```
Int N=0, k=N;
While (k>0)
{
        Printf (“%d\n”, k);
        k/=2;
}
Printf (“final= %d\n”, k);
```

4. Indique los valores que toman las variables i, j. para N = 0, 1, 10, 20, 40, 60, 80, 100 en los siguientes ciclos de carrera:

```
a.
int i=0, j=0;
for (i=0, j=0; i<N && j<N; i+=6, j+=6)
   printf("%d\n", k);
printf("final= %d", k);

b.
int i=0, j=0;
for (i=0, j=0; i<N || j<N; i+=6, j+=6)
   printf("%d\n", k);
printf("final= %d", k);

c.
int i=0, j=0;
for (i=N, j=0; i>0 && j<N; i-=6, j+=6)
   printf("%d\n", k);
printf("final= %d", k);

d.
int i=0, j=0;
for (i=N, j=0; i>0 || j<N; i-=6, j+=6)
   printf("%d\n", k);
printf("final= %d", k);
```

a. Para N=0, no se cumple la condición, así que i=0, j=0.
Para N=1, la condición se cumple, así que i=6, j=6; la condición ya no se cumple.
Para N=10, la condición se cumple, el primer bucle se obtiene i=6, j=6; en la segunda iteración i=12, j=12.
Para N=20, el bucle se realiza hasta 4 veces, entonces i=24, j=24.
Para N=40, el bucle se realiza hasta 7 veces, entonces i=42, j=42.
Para N=60, el bucle se realiza hasta 10 veces, i=60, j=60.
Para N=80, el bucle se repite hasta 14 veces, i=84, j=84.
Para N=100, el bucle se repite hasta 17 veces, i=102, j=102.

b. Para N=0, no se cumple la condición la condición, i=0, j=0.
Para N=1, se cumple la condición, el bucle de repite 1 vez, i=6, j=6.
Para N=10, el bucle se realiza hasta 2 veces, i=12, j=12.
Para N=20, el bucle se realiza hasta 4 veces, i=24, j=24.
Para N=40, el bucle se repite hasta 7 veces, i=42, j=42.
Para N=60, el bucle se repite hasta 10 veces, i=60, j=60.
Para N=80, el bucle se realiza hasta 14 veces, i=84, j=84.
Para N=100, el bucle se realiza hasta 17 veces, i=102, i=102.

c. Para N=0, i no es mayor a 0 y j no es menor a N (0), por tanto, i=0, j=0.
Para N=1, i es mayor a 0 y j sí es menor a 1, por tanto, i=-5, j=6.
Para N=10, i es mayor a 0 y j es menor a 10, por tanto, i=4, j=6. Se sigue cumpliendo la condición, por lo que i=-2, j=12. Se deja de cumplir la condición.
Para N=20, i es mayor a 0 y j es menor a 20, por tanto, realizando todas las iteraciones hasta que se deje de cumplir la condición, i= -4, j=24.
Para N=40, realizando todas las iteraciones hasta que se deje de cumplir la condición, i=-2, j=42.
Para N=60, cuando se deja de cumplir la condición, i=0, j=60.
Para N=80, la condición no se cumple cuando i=-4, j=84.
Para N=100, la condición ya no se cumple si i=-2, j=102

d. Para N=0, ninguna de las condiciones se cumple, por tanto, i=0, j=0.
Para N=1, i es mayor a 0, también j es menor a 1, entonces, i=-5, j=6.
Para N=10, i es mayor a 0, también j es menor a 10, por tanto, repitiendo el bucle una vez más, i=-2, j=12.
Para N=20, i es mayor a 0, j es menor a 20, repitiendo el bucle se detiene cuando i=-4, j=24.
Para N=40, el bucle acaba cuando i=-2, j=42.
Para N=60, el bucle termina cuando i=0, j=60.
Para N=80, el bucle finaliza si i=-4, j=84.
Para N=100, se deja de iterar cuando i=-2, j=102.

5. Indique la salida de los siguientes ciclos anidados para L, N, M = 1, 2, 3, 5,7,11,13:

```
a.
int i=0, j=0, k=0;
for (i=0; i<N; i++)
   for (j=0; j<M; j++)
      printf("i=%d j= %d\n");
printf("i=%d j= %d\n");

b.
int i=0, j=0;
for (i=0; i<N; i++)
   for (j=0; j<M; j++)
      for (k=0; k<L; k++)
         printf("i=%d j= %d\n");
printf("i=%d j= %d\n");
```

a. Para todo valor de N, M mayor a 0:
Se inicia imprimiendo i=0 j=0, y j va aumentando en 1 hasta alcanzar M-1; cuando suceda esto, j se resetea hasta 0 e i aumenta en 1; si la condición lo permite, se vuelve a imprimir los valores, pero ahora empezando con el nuevo valor de i hasta que j alcance M-1, y así se seguirá sumando i hasta que llegue a N-1. Si la condición no permite cierto valor de i, se termina todos los bucles. Al final se imprime i=N j=M.

b. Para todo valor de N, M, L mayor a 0:
Se inicia imprimiendo i=0 j=0 L veces, pero cuando el bucle de L finaliza, pasa a que j aumenta en 1 y se repite la repetición de L veces. Cuando j alcanza su valor máximo M-1, éste se reinicia a 0 e i aumenta en 1 y, si la condición lo permite, se vuelve a realizar las impresiones desde j=0 L veces hasta que ésta termine; todo el proceso inicial, pero con diferente valor, se volverá a realizar hasta que i alcance el

valor de N-1. Si la condición no permite el valor de i, todos los bucles se terminan. Cuando acaban todos los bucles, se imprime i=N j=M.

6. ¿Cuál es la salida del siguiente código?

```
void main()
{
      int u=0, v=0, w=0, x=0, y=0, z=0;
      for (u= 0; u<=1; u++)
         for (v= 0; v<=1; v++)
            for (w=0; w<=1; w++)
               for (x=0; x<=1; x++)
                  {
                    y= convertir(u, v, w, x);
                    z= convertir(x, w, v, u);
                  }
}

 int convertir(int a, int b, int c, int d)
 {
    int e=0;
    e= 8*a + 4*b + 2*c + 1*d;
    a= 1;   b= 2;   c= 3;   d=4;
    return(e);
 }
```

Cuando las variables u, v, w, x tienen valor de 0, y & z son 0. Cuando x es 1 y los demás 0, y=1 & z=8. Con w=1, x=0 y los demás 0, y=2 & z=4. Con w=1 & x=1, y=3 & z=12. Teniendo estos primeros datos, es posible deducir la salida de las demás variables, solo habría que sumarle a y & z lo que se obtenga con el nuevo valor de las variables.

Con v=1, w=0 & x=0, y= 4 & z=2; con los valores de y & z con v=1, podemos obtener los demás valores cuando las otras variables son 1 o 0 gracias a que se calculó anteriormente. Cuando w=0 & x=1, y=5 & z=10; cuando w=1 & x=0, y=6 & z=6; con w=1 & x=1, y=7 & z=14.

Con u=1, v, w, x siendo 0, y=8 & z=1; con lo conseguido en v=1, solo sería necesario calcular la suma de y & z (con distintos valores de v, w, x) con lo obtenido cuando u=1. Manteniendo v=0 y variando w=0 & x=1, y=9 & z=9; cuando w=1 & x=0, y=10 & z=5; cuando w=1 & x=1, y=11 & z=13. Ahora, manteniendo v=1 y variando w=0 & x=0, y=12 &

z=3; cuando w=0 & x=1, y=13 & z=11; cuando w=1 & x=0, y=14 & z=7; cuando w=1 & x=1, y=15 & z=15.

Estos resultados son los equivalentes binarios de los valores de u, v, w, x & x, w, v, u, de y & z respectivamente.

7. Codifique un programa para calcular la energía cinética K, potencial U, el Hamiltoniano $H = K + U$ y el Lagrangiano $L = K - U$, en términos del tiempo t = 0,1, 2, …, N, de un objeto que se encuentra en las siguientes situaciones físicas (Nota: escriba una función para cada término energético, K, U, H y L):
   a. Caída libre, el objeto se detiene cuando la energía potencial es igual a cero:
      $K = \frac{1}{2}mv^2$, $U = mgh$
      $v = v_o - gt$
      $h = h_0 + v_0t - \frac{1}{2}gt^2$
   b. Ascenso, el objeto se detiene cuando la energía cinética es igual a cero:
      $K = \frac{1}{2}mv^2$, $U = mgh$
      $v = v_o - gt$
      $h = h_0 + v_0t - \frac{1}{2}gt^2$
   c. Movimiento circular uniforme:
      $K = \frac{1}{2}mv^2$, $U = kx^2$
      $x = A\cos(\omega t + \phi)$
      $v = -A\,\omega\sin(\omega t + \phi)$

De forma general, tenemos que el valor de g es 9.8, pero dependiendo de la situación física ésta es positiva o negativa (caída libre y ascenso, respectivamente). Además, m es la masa; v, la velocidad; v_0, la velocidad inicial; h, la altura; h_0, la altura inicial; t, el tiempo.

Por otra parte, en las fórmulas del movimiento circular uniforme, la fórmula de energía cinética es la misma, pero la energía potencial cambia con k & x, siendo k el coeficiente de resorte, x & v pertenecen al movimiento armónico simple, en donde x es la posición del móvil que describe el movimiento y v es la velocidad; A es la amplitud; ω, la frecuencia angular; $\omega t + \phi$, la fase; ϕ, la fase inicial.

   a. Caída libre. Sustituyendo la velocidad v en K, se obtiene $K = \frac{1}{2}m(v_o - gt)^2$; para U, se sustituye la altura h en U, dando $U = mg(h_0 + v_0t - \frac{1}{2}gt^2)$; por lo tanto, el Hamiltoniano es $H = K + U = \frac{1}{2}m(v_0 - gt^2) + h_0 + v_0t - \frac{1}{2}gt^2$, y el Lagrangiano es $L = K - U = \frac{1}{2}m(v_0 - gt^2) - \left(h_0 + v_0t - \frac{1}{2}gt^2\right) = \frac{1}{2}m(v_0 - gt^2) - h_0 - v_0t + \frac{1}{2}gt^2$.

b. Ascenso. Para la energía cinética, se sustituye la velocidad v en K, dando: $K = \frac{1}{2} m(v_o - gt)^2$; para la energía potencial, se sustituye la altura h en U, obteniendo: $U = mg(h_0 + v_0 t - \frac{1}{2} gt^2)$; el Hamiltoniano es $H = \frac{1}{2} m(v_0 - gt^2) + h_0 + v_0 t - \frac{1}{2} gt^2$, y el Lagrangiano es $L = \frac{1}{2} m(v_0 - gt^2) - h_0 - v_0 t + \frac{1}{2} gt^2$. Como se puede apreciar, las fórmulas son las mismas, sin embargo, el signo de la gravedad es negativa para este caso; para el primer caso, la gravedad es positiva.

c. Movimiento circular uniforme. Para la energía cinética, se sustituye x en K, teniendo: $K = \frac{1}{2} m\left(-A\omega sin(\omega t + \phi)\right)^2$; para la energía potencial, se sustituye v en U, dando: $U = k(Acos\,(\omega t + \phi))^2$. El Hamiltoniano es $H = \frac{1}{2} m\left(-A\omega sin(\omega t + \phi)\right)^2 + k(Acos\,(\omega t + \phi))^2$; el Lagrangiano es $L = \frac{1}{2} m\left(-A\omega sin(\omega t + \phi)\right)^2 - k(Acos\,(\omega t + \phi))^2$.

8. ¿Cuál es el contenido del arreglo al final de los siguientes ciclos?

```
int arr[100];
int k=0, c=0;
```

a.
```
for (k=0; k<N; k++)
   arr[k]= 2*k;
```

c.
```
for (k=0, c=1; k<N && c<N; k++, c++)
   {
     arr[k]= 2*k;
     arr[c]= 0;
   }
```

b.
```
for (k=0; k<N; k++)
   arr[k]= 2*k+1;
```

d.
```
for (k=0, c=1; k<N && c<N; k++, c++)
   {
     arr[k]= 0;
     arr[c]= 2*k+1;
   }
```

a. El arreglo tendrá como contenido final una sucesión de números pares empezando desde 0 hasta 2*(N-1). Debido a que en k=N el ciclo acaba, k=N-1 es el último valor válido.

b. El contenido del arreglo al final será una sucesión de números impares iniciando desde 1 hasta 2*(N-1) +1. Debido a que en k=N el ciclo termina, k=N-1 es el último valor válido.

c. El contenido del arreglo al final del ciclo será una sucesión de números pares partiendo de 0 hasta 2*(N-2); debido a que c vale 1, llega a N antes que k; k solo alcanza el valor de N-2, mientras que c alcance N-1 (esto antes de c=N), y cuando c sea igual a N, el AND dejará de cumplirse.

d. Debido a que en todos los valores de k se le asigna el 0, entonces, el contenido del arreglo inicia con una sucesión de 0 hasta un número impar 2*(N-1)+1; esto último

no es 0 pues c adelanta a k en 1 y, debido a esto, c llega a N antes que k, lo que permite que en la posición c, o N-1, se le pueda asignar un número impar.

9. Calcule el número decimal correspondiente al arreglo de bits (el dígito menos significativo es el $digito$[0]):

```
int N= 8;
float digito[N]= {1, 0, 0, 1, 0, 1, 0, 1};
float num=0.0, k=0.0;

for (k=0; k<N; k++)

   num+= digito[k]*pow(2, k);
```

Modifique el código para que pueda convertir de cualquier base a decimal y calcule el número correspondiente para convertir los siguientes casos: a) base 4: {3,2,0,2,1,1,0,3}, b) base 16: {5, E, 9, C, 2, 3, F, A} y base 20 {15, 9, 7,1,3,10,5,2}. Escriba el código respectivo para convertir de nuevo el número decimal a otra base distinta.

Como el digito menos significativo es digito[0], entonces a este le corresponde 2^0, en N=1 se encuentra 2^1, y así hasta N=7 con 2^7. "Activando" las casillas 0, 3, 5 y 7 del arreglo, y sumando sus contenidos, se obtiene: 169.

Para convertir cualquier número de base n de N=8 dígitos a decimal, se usaría un arreglo de longitud N. Para el caso de un número alfanumérico, como la base 16, habría que convertir las letras a su valor numérico, siendo sus valores: A=10; B=11; C=12; D=13; E=14; F=15; además, en caso de ser mayor a la base 16, la sucesión de letras continuaría hasta la Z. Una vez sabiendo qué valor tienen las letras, el arreglo entra en un ciclo en el que se multiplica el contenido de la casilla por la base n elevado a la potencia del número de casilla, comenzando en 0 y terminando en N-1. Este número multiplicado se irá sumando con los próximos productos de la base n. Esta suma sería el número en decimal.

a) El digito menos significativo es 3, esto se multiplica por 4 elevado a la 0 debido a su posición. Esto conlleva a que la siguiente casilla digito[1] es 2 y se multiplicará por 4 elevado a la posición de la casilla; por lo tanto, num= $3*4^0 + 2*4^1 + 0*4^2 + 2*4^3 + 1*4^4 + 1*4^5 + 0*4^6 + 3*4^7$ = 3+8+0+128+256+1024+0+49152 = 50,571.
b) Sabiendo las equivalencias en hexadecimal, tenemos num=$5*16^0 + 14*16^1 + 9*16^2 + 12*16^3 + 2*16^4 + 3*16^5 + 15*16^6 + 10*16^7$ = 2,939,341,285.
c) Las cifras en base 20 también pueden estar en una representación con letras como en hexadecimal, de esa forma no se confundirán las cifras. Tenemos que num= 15(=F) $*20^0 + 9*20^1 + 7*20^2 + 1*20^3 + 3*20^4 + 10$(=A) $*20^5 + 5*20^6 + 2*20^7$ = 2912490995.

10. Escriba un programa para generar las siguientes secuencias y almacenarlas en un arreglo de 100 números (en caso de desborde, utilice una sentencia break para terminar el ciclo):
    a. 1,4,7,10,13,16,19,22,25, …
    b. 2,4,8,16,32,64,128,256, …
    c. 1,3,6,10,15,21,28,36,45, …
    d. 1,4,9,16,25,36,49,64,81, …
    e. La secuencia de Fibonacci
    f. Los números primos

    En un caso general, se utiliza un contador k, y un arreglo de longitud 100. Para que el bucle siga iterando, se tendrá la condición de que k sea menor a 100.

    a) Iniciamos un bucle en el que k=0 irá aumentando en una unidad cada vez que acabe una iteración. Durante la iteración, habrá una condición que comprobará que 3*k+1 sea menor a un determinado número dependiendo del tipo de variable; en caso de que 3*k+1 sea mayor, se detendrá el bucle; en caso contrario, continuará con lo siguiente: se le asignará a la posición k del arreglo el resultado de 3*k+1; estas operaciones se repetirán hasta que k alcance el valor de 100 y, entonces, el bucle terminará.

    b) Empezamos un bucle con dos contadores, k=0 & c=1, el cual aumenta en una unidad cada vez que acaba una iteración. La primera instrucción sería una condición en la que se comprobaría que 2*c sea menor a un determinado número dependiendo del tipo de variable; en caso de que 2*c sea mayor al número, se detendría el bucle; en caso contrario, a la posición k del arreglo se le asignará el resultado de 2*c; estas operaciones se repetirán hasta que k alcance el valor de 100 y, entonces, el bucle terminará.

    c) Declaramos una variable aux con valor de 1 y empezamos un bucle con dos contadores, k=0 & c=2, los cuales aumentan en una unidad cada vez que acaba una iteración. La primera instrucción sería una condición en la que se comprobaría que aux sea menor a un determinado número dependiendo del tipo de variable; en caso de que aux sea mayor al número, se detendría el bucle; en caso contrario, a la posición k del arreglo se le asignará el valor de aux. Después, la variable aux obtiene un nuevo valor, siendo este aux+c; luego, se repiten estas iteraciones hasta que k alcance el valor de 100 y el bucle terminará.

    d) Declaramos una variable aux con valor de 1 y empezamos un bucle con dos contadores, k=0 & c=3, los cuales aumentan en una y dos unidades, respectivamente, cada vez que acaba una iteración. La primera instrucción sería una condición en la que se comprobaría que aux sea menor a un

determinado número dependiendo del tipo de variable; en caso de que aux sea mayor al número, se detendría el bucle; en caso contrario, a la posición k del arreglo se le asignará el valor de aux. Después, la variable aux obtiene un nuevo valor, siendo este aux+c; luego, se repiten estas iteraciones hasta que k alcance el valor de 100 y el bucle terminará.

e) Establecemos que en la posición 0 y 1 del arreglo estarán los números 0 y 1. En el bucle, tendremos un contador k=2; la primera instrucción sería una condición en la que se comprobaría que la suma de la casillas k-1 y k-2 sea menor a un determinado número dependiendo del tipo de variable; en caso que la suma de las casillas k-1 y k-2 sea mayor al número, se detendría el bucle; en caso contrario, a la posición k del arreglo se le asignará el resultado de la suma de las casillas k-1 y k-2. Se repetirá este procedimiento hasta que k alcance 100, y luego el bucle terminará.

f) Se utiliza dos contadores en un bucle, k=0 y num=1, los cuales aumentarán en una unidad cada vez que termina una iteración. Además, dentro del bucle, se tendrán dos variables, aux=1 y c=2. k será la posición de casilla y num el número que será asignado a esa casilla. Primero, se comprobará que num no supere un determinado número, el cual depende del tipo de variable; en caso de que num sea mayor, el bucle terminará; en caso contrario, se hará lo siguiente: en el bucle se tendrá un ciclo que verificará que se cumpla la condición de "c es menor o igual a la mitad de num y que la variable aux siga siendo 1". Si esto se cumple, se comprueba que el residuo de la división de num y c sea 0, si es así, aux será 0; luego, el contador c aumenta en 1 y se repite la verificación. Si no se cumple la condición del ciclo, se pasa a comprobar una condición de que la variable aux sea 1; si es así, en la casilla k del arreglo se le asignará el valor de num. En caso contrario, se le restará en una unidad a k, de esta forma la posición seguirá siendo la misma en caso de que no se cumpla la condición. Este procedimiento se repite hasta que k alcance el valor de 100 y, entonces, el bucle terminará.

11. Codifique un programa que encuentre todos los factores primos de los números que se encuentran entre 1 y N. Pruebe su programa para N= 100, 1000, 10000, 100000, 1000000.

Se requerirá de dos arreglos, uno que contenga tantos números primos como sean necesarios (el último número primo del arreglo puede ser uno que esté cercano a N), y otro que almacenará los factores primos de N.

Se utilizará un bucle en el cual se irá dividiendo cada número primo del primer arreglo entre N, y el resultado de la división se almacenará en el segundo arreglo. Para que la división de cada número primo se pueda realizar habrá una condición

que comprobará que la división dé residuo 0, así se dividirá cuantas veces sea necesario hasta que la división dé un residuo; cuando ocurra esto, se cambiará de número primo.

Para conocer cuáles números son factores primos de N, se imprimirá el contenido del arreglo.

Nota: Por defecto, la casilla de la posición 0 del segundo arreglo será 1.

Tendremos que los factores primos son 1, 2 y 5; el producto de estos es 10, pero si lo elevamos a la potencia 2, 3, 4, 5 y 6 se obtendrá cada valor de N respectivamente.

12. Codifique un programa que a) encuentre el área y el perímetro de un polígono regular de N lados. La única información con la que cuenta es el número de lados N, y la longitud a del lado. Alternativamente, b) encuentre el área y el perímetro del polígono regular de N lados, si desconoce la longitud de cada lado, pero sabe que se encuentra inscrito en un círculo de radio R. Previamente realice el análisis geométrico y algebraico y escriba las expresiones analíticas para calcular el área y el perímetro, en función de N y a: $area(N, a)$ y $perimetro(N, a)$; y en función de N y R: $area(N, R)$ y $perimetro(N, R)$.

    a) Dado un lado N de un polígono regular, es posible obtener los datos necesarios para hallar el área. El polígono regular de N lados puede ser partido en N triángulos isósceles, con esto se puede obtener un ángulo de cada triangulo dividiendo 360°/N, este sería el ángulo A; sabiendo que la suma de todos los ángulos internos de un triángulo es 180°, podemos llegar a que el ángulo B está dado por 180° - A y el resultado de la resta se divide entre 2, puesto que son dos ángulos iguales y la suma de estos dos con A es 180°. Con esto tenemos los tres ángulos.

    Ahora, necesitamos la altura h del triángulo, el cual funcionará de apotema, pero para eso se requiere la hipotenusa H. Para ello, el triángulo isósceles hay que convertirlo en un triángulo rectángulo, basta con sólo dividir la longitud 'a' y el ángulo A por la mitad. Usando la razón trigonométrica coseno podemos obtener la hipotenusa: cos(B)=(a/2) /H -> H=(a/2) /cos(B); ya con la hipotenusa, usamos el teorema de Pitágoras para hallar la altura. Inicialmente se tienen: $H = \sqrt{\left(\frac{a}{2}\right)^2 + h^2} \rightarrow h = \sqrt{H^2 - \left(\frac{a}{2}\right)^2}$
    Con estos datos, el área se puede calcular con la fórmula del área del triángulo isósceles y multiplicándolo por la cantidad de lados del polígono:
    Área=N*(a*h) /2.
    Para el cálculo del perímetro, se multiplica la longitud del lado por la cantidad de lados: P=N*a.

b) Teniendo solamente el radio R, el procedimiento es similar al anterior; el polígono se divide en N triángulos isósceles, para luego hacerlo un triángulo rectángulo dividiendo por la mitad el triángulo, teniendo que la base sea a/2 y el ángulo C sea (Angulo A/2); obtenemos la medida de la base con la razón trigonométrica seno: sen(C)= (a/2)/R -> a= 2*R*sen(C). Después, utilizando el teorema de Pitágoras, obtendremos la altura h:
$R = \sqrt{a^2 + h^2} \rightarrow h^2 = R^2 - a^2 \rightarrow h = \sqrt{R^2 - a^2}$.
Con estos datos, para calcular el área, volvemos a usar la fórmula del área del triángulo y lo multiplicamos por la cantidad de lados N: Área=N*(a*h) /2.
Para el perímetro, simplemente multiplicamos la cantidad de lados por la base: P=N*a.

13. Considere que cuenta con un arreglo unidimensional de N enteros aleatorios. Codifique un programa que realice las siguientes actividades:
    a. Encontrar el número más grande y el número más pequeño de todos.
    b. Calcular el valor promedio y la desviación estándar de todos los números.
    c. Ordenar los números de forma ascendente y descendente.
    d. Desordenar aleatoriamente el arreglo.

a) Para encontrar el número más grande se hace utilizando una variable auxiliar, un bucle y una condición; se hará lo siguiente: antes de comenzar con el bucle, se le asignará a la variable auxiliar el valor de 0; el bucle empezará con un contador k=0 mientras k sea menor a N, donde N es la cantidad de enteros en el arreglo; además, al final de cada iteración, k se sumará en una unidad. Dentro del bucle, habrá una condición que comprobará si el valor de la variable auxiliar es menor que el número en la posición k del arreglo; si esto es verdadero, entonces la variable auxiliar adquiere el número de la posición k del arreglo y el bucle continuaría y se seguiría realizando la misma condición; en caso de no cumplirse, el bucle continúa y se sigue realizando la condición, en cualquiera de los dos casos, el bucle acabará cuando k alcance N. Cuando el bucle acabe, el valor final de la variable auxiliar es el número más grande entre todos del arreglo.

Para el número más pequeño, sólo hay que cambiar la condición que está dentro del bucle; comprobará si el valor de la variable auxiliar es mayor que el número en la posición k del arreglo; si esto se cumple, entonces la variable auxiliar adquiere el valor del número de la posición k del arreglo y continuaría el bucle y se seguiría realizando la misma condición; y en caso de no cumplirse, el bucle continuaría y se seguiría efectuando la misma condición; para cualquiera de los dos casos, el bucle terminará cuando k alcance N. Cuando esto suceda, el valor final de la variable auxiliar es el número más pequeño entre todos del arreglo.

b) Con una variable prom se almacenará la división entre N de la suma de los N números, con una variable suma se realizará la suma de los N números, y se tendrá un bucle que realizará la sumatoria; suma comenzará con valor de 0. El bucle contará con un contador k=0 y seguirá mientras k sea menor a N y al final de cada iteración k aumentará en uno. Suma obtendrá el valor de la suma entre suma y el número en la posición k del arreglo, luego, se hará otra iteración en donde se seguirá haciendo la misma suma hasta que k alcance N. Terminando el bucle, prom adquiere el valor del resultado de la división entre suma y N.

   Para la desviación estándar, se empleará una variable para la desviación estándar, para la sumatoria, con valor a 0, y la misma variable del promedio, además de un nuevo arreglo. Iniciamos un bucle con un contador i=0 y realizará operaciones mientras i sea menor a N, e i aumentará en una unidad cada vez que finalice una iteración. La casilla i del nuevo arreglo almacenará la resta entre el número de la posición i del arreglo de enteros aleatorios y el promedio, y este resultado se elevará al cuadrado. Al finalizar el bucle, se inicia otro bucle con un contador j=0, e irá iterando mientras j sea menor a N, además que después de cada iteración aumenta en uno. Este bucle se encargará de la sumatoria, por lo que la variable sumatoria obtendrá el valor de sumarse a sí misma y al contenido del nuevo arreglo en la posición j; la suma se repetirá hasta que se deje de cumplir la condición de j menor a N. Ya teniendo la sumatoria, la desviación estándar está dada por la raíz cuadrada de la sumatoria de las restas al cuadrado dividido entre N-1. El resultado es la desviación estándar.

c) Para ordenar de forma ascendente, emplearemos una variable aux y dos bucles; el primer bucle contará con un contador i=0, mientras i sea menor a N, i aumentará en uno cada que vez finalice una iteración. Dentro de este bucle, habrá otro con un contador j=i+1 (esto para que compruebe el valor siguiente de i), mientras j sea menor a N, j aumentará en uno al acabar cada iteración; en este segundo bucle, habrá una condición que comprobará que el número del arreglo en la posición j sea menor al número del mismo arreglo en la posición i. Si esto se cumple, el número de la posición j intercambia de valor con el de la posición i, es decir, aux tomará el valor del número de la posición j del arreglo, luego la casilla j obtendrá el valor de la casilla i del arreglo, después en la casilla i se tendrá como nuevo valor el contenido de aux. Si esto no se cumple, entonces se vuelve a hacer el segundo bucle con el valor aumentado de j. Cuando j alcance el valor de N, se terminará su bucle e i aumentará en uno, para después volver a iniciar el bucle de j. Esto se repetirá hasta que i alcance el valor de N, cuando esto suceda, el bucle se detendrá.
   Para la forma descendente, es similar a la forma ascendente, con la única diferencia de que en la condición del segundo bucle se comprobará que la posición

j del arreglo sea mayor a la posición i del mismo arreglo. La asignación de variables es la misma, al igual de la forma en que acabarán ambos bucles.

d) Tendremos tres arreglos, uno con los enteros aleatorios, otro que almacenará estos enteros en un orden aleatorio y otro que se encargará que no se repitan las posiciones aleatorias, este es el arreglo de repeticiones (en la casilla 0 almacenará el número 0). Dentro de un bucle con un contador i=0, mientras i sea menor a N, i aumentará en uno cada vez que acabe una iteración; a una variable k se le asignará un valor aleatorio entre 0 y N, luego se iniciará otro bucle con un contador j=0, mientras j sea menor o igual a i, j aumentará en uno cada que vez finalice una iteración.
Dentro de este segundo bucle, habrá una condición que comprobará que la posición 0 del arreglo de repeticiones sea igual a k=0, si es así, entonces en el arreglo de orden aleatorio se almacenará el número de la posición k del arreglo de enteros. Si no, pasará a otra condición en la que se comprobará que la posición j del arreglo de repeticiones no sea igual al número aleatorio de k, en caso de que no sean iguales, en el arreglo de orden aleatorio en su casilla i, será almacenado el número de la posición k del arreglo de enteros, y se guardará el número de k en la casilla i+1 del arreglo de repeticiones, de esa forma se evitará que se repitan los números cuando no se cumpla la condición. En caso de que sí sean iguales, entonces la casilla i del arreglo de orden aleatorio será 0, en la posición i+1 del arreglo de repeticiones será 0, ya que no se aceptan números repetidos, e i se restará en una unidad para que se repita el ciclo con las mismas casillas, pero con un diferente valor de k.
Estas operaciones se repetirán hasta que i llegue al valor de N y se detendrá el bucle; para comprobar que sí se ordenaron de manera aleatoria, se imprimen ambos arreglos.

14. Considere que cuenta con un arreglo unidimensional de caracteres, de longitud arbitraria, pero cuyo final de cadena esta denotado por el carácter especial '\0'. Codifique un programa que realice las siguientes actividades:
    a. Encontrar la longitud de la cadena.
    b. Ordenar alfabéticamente, en orden ascendente y descendente, las letras de la cadena.
    c. Desordenar aleatoriamente los caracteres de la cadena.
    d. Invertir el orden de los caracteres de la cadena.
    e. Cambiar las letras de mayúsculas a minúsculas y viceversa.
    f. Insertar un espacio entre dos caracteres (Nota: todos los caracteres originales deben estar presentes al final del espaciado, esto causará que la longitud de la cadena aumente).

a) En un bucle con un contador i=0, en donde se seguirá repitiendo las operaciones mientras la casilla i del arreglo no sea igual al final de cadena '\0', y al final de cada iteración i aumentará en uno; en el bucle no se realizará ninguna operación. En la finalización del bucle, el número que alcanzó el contador i es la longitud de la cadena.

b) Para el ordenamiento ascendente, se usará una variable aux de tipo cadena y dos bucles; el primer bucle tendrá un contador i=0, mientras que en la casilla i de la cadena no se encuentre el '\0', y al final de cada iteración i aumenta en uno. Dentro de este bucle, habrá otro con un contador j=i+1 (esto para que compruebe el valor siguiente de i), mientras que en la casilla j del arreglo no se encuentre un '\0', y j aumentará en uno al acabar cada iteración; en este segundo bucle, habrá una condición que comprobará que la letra del arreglo en la posición j sea menor a la letra del mismo arreglo en la posición i. Si esto se cumple, la letra de la posición j intercambia de letra con la de la posición i, es decir, aux tomará el carácter de la posición j del arreglo, luego la casilla j obtendrá el carácter de la casilla i del arreglo, después en la casilla i se tendrá como nuevo carácter el contenido de aux. Si esto no se cumple, entonces se vuelve a hacer el segundo bucle con el valor aumentado de j. Cuando la casilla j se encuentre con '\0', se terminará su bucle e i aumentará en uno, para después volver a iniciar el bucle de j. Esto se repetirá hasta que en la casilla i se encuentre con '\0', cuando esto suceda, el bucle se detendrá.
Para la forma descendente, es similar a la forma ascendente, con la única diferencia de que en la condición del segundo bucle se comprobará que la posición j del arreglo sea mayor a la posición i del mismo arreglo. La asignación de caracteres es la misma, al igual de la forma en que acabarán ambos bucles.

c) Se utilizará un arreglo numérico para guardar las posiciones repetidas, y otro arreglo de caracteres para la cadena de orden aleatorio. En un bucle de contador i=0, mientras i sea menor a la longitud de la cadena, i aumentará en uno al final de cada iteración; dentro de este bucle, habrá una variable k que almacenará un número aleatorio entre 0 y l, luego se iniciará otro bucle con un contador j=0, mientras j sea menor o igual a i, j aumentará en uno al final de cada iteración. Dentro de este segundo bucle, habrá una condición que comprobará que la posición 0 del arreglo de repeticiones sea igual a k=0, si es así, entonces en el arreglo de orden aleatorio se almacenará la letra de la posición k del arreglo de la cadena. Si no, pasará a otra condición en la que se comprobará que la posición j del arreglo de repeticiones no sea igual al número aleatorio de k, en caso de que no sean iguales, en el arreglo de orden aleatorio, en su casilla i, será almacenado la letra de la posición k del arreglo de la cadena, y se guardará el número de k en la casilla i+1 del arreglo de repeticiones, de esa forma se evitará que se repitan los números cuando no se cumpla la condición. En caso de que sí sean iguales,

entonces la casilla i del arreglo de orden aleatorio será una cadena vacía, en la posición i+1 del arreglo de repeticiones será 0, ya que no se permiten números repetidos, e i se restará en uno para que se repita el ciclo con las mismas casillas, pero con un diferente valor de k.

d) En un arreglo que llamaré alfa guardaré la cadena invertida y se utilizará la longitud de la cadena. En un bucle de contador i=0, mientras i sea menor a la longitud de la cadena, i aumentará en uno al final de cada iteración; dentro del bucle, la casilla i del arreglo alfa será la casilla l-(i+1) del arreglo de la cadena. La razón de l-(i+1) es porque en la casilla l estamos tomando el final de la cadena, es decir, '\0' y no queremos que el final de cadena esté al principio, por lo que para evitarlo sería utilizar la casilla anterior a l, es decir, l-1; pero debido a que estaremos recorriendo la cadena desde el final hasta el comienzo, i puede ayudar a recorrer las casillas necesarias; por eso es l-(i+1). Después de que finalice el bucle, le añadimos el final de cadena en la casilla l.

e) Para convertir a mayúsculas: en un bucle con un contador i=0, mientras la casilla i no se encuentre con el final de cadena, i aumentará en una unidad. Dentro de este bucle se hallará una condición que comprobará que la posición i del arreglo de la cadena sea mayor a 96 (en ASCII la letra 'a' es representada por 97, así que se inicia desde aquí), si esto es verdadero, entonces la posición i del arreglo de la cadena se le asignará la letra representada por el carácter de esa casilla i menos 32 ('A' es el 65 en ASCII y entre 'a' y 'A' hay una diferencia de 32). Si la condición no se cumple, entonces no se hace nada a esa posición del arreglo. Y el bucle seguirá hasta que se encuentre el fin de cadena; cuando esto ocurra, el bucle acabará.

   Para convertir a minúsculas, sería el mismo procedimiento que el anterior, pero esta vez la condición comprobará que la casilla i del arreglo de la cadena sea mayor a 64 (esto por lo ya mencionado del ASCII) y también que esta misma casilla sea menor a 91, así no va a considerar los caracteres superiores a ese número y no solo las letras minúsculas. Además, en lugar de que se reste 32, se le sumará 32 a la letra en la posición i. Si la condición no se cumple, entonces no se hará nada en esa posición del arreglo.

f) Emplearé un arreglo de espaciado para almacenar la cadena con espacios entre dos caracteres. En un bucle con contador i=0, mientras la casilla i del arreglo de la cadena no se encuentre con el final de cadena, i aumentará en uno al final de cada iteración; dentro del bucle, en la casilla 2*i del arreglo de espaciado se le asignará el contenido de la casilla i del arreglo de la cadena. Luego, en la casilla 2*i+1 del arreglo de espaciado, tendrá como contenido un espacio ' '; la razón de 2*i es porque en las casillas impares del arreglo de espaciado estarán los espacios, por

ejemplo, teniendo dos objetos cualesquiera A y B, y se coloca un objeto C entre estos dos, este objeto C estaría en una posición impar, ya que A estaría en posición 0 y B en la posición 2, por lo que C está en 1. Estas operaciones se repetirán hasta que se llegue al final de cadena y, por ende, acabará el bucle.

Al final de todas las operaciones, las cadenas resultantes deben encontrarse delimitadas al final por el carácter de fin de cadena '\0'.

15. ¿Cuál es el contenido de la matriz para el par de ciclos dobles?

```
int matriz[10][10];
int i=0, j=0, N=10;

for (i=0; i<N; i++)
   for (j=0; j<N; j++)
      matriz[i][j]= i*j;

for (i=0; i<N; i++)
   for (j=0; j<N; j++)
      matriz[i][j]= i+j;
```

En el primer ciclo doble, la fila 0 estará llena de 0's, la fila 1 será una sucesión del 0 al 9, la fila 2 es una sucesión de números pares del 0 al 18, la fila 3 será una sucesión de múltiplos de 3 del 0 al 27. Con lo anterior, se intuye que cada fila de la matriz es la tabla de multiplicación del valor de la fila, empezando desde 0 hasta i*9.

En el segundo ciclo doble, se tiene que la fila 0 es una sucesión del 0 al 9, pero en la fila 1 es una sucesión del 1 al 10, y en la fila 2 hay una sucesión del 2 al 11. Por lo tanto, en cada fila se empieza con el número de esa misma fila, y continúa la sucesión de uno en uno hasta el final de la fila, en donde estará el número i+9 para cada fila.

16. ¿Cuál es el contenido del arreglo al final del siguiente código?

```
int matriz[8][8];
int pieza[8]= {1, 4, 2, 3, 7, 0, 6, 5};
int i=0, j=0, k=0, N=8;

for (i=0; i<N; i++)
   for (j=0; j<N; j++)
      {
        if (i%2==0)
          matriz[i][j]= j%2;
        else
          matriz[i][j]= (j+1)%2;
      }

for (i=0, k=N; i<N; i++, k--)
   {
     j= pieza[k];
     matriz[i][j]= k;
   }
```

En el primer ciclo se tiene que en cada fila par se inicia con un 0, en la casilla siguiente habrá un 1, en el siguiente un 0 y así sucesivamente, y el final de la fila, en la octava casilla, sería un 1. En las filas impares se inician con un 1 y en la siguiente casilla habría un 0, después un 1 y así sucesivamente hasta que en la octava casilla de la fila haya un 0. Esto es así para todas las filas.

El segundo bucle nos cambia algunos números de la matriz. En la fila 0 y columna j=5, habrá un 8 (esto siendo el valor de k, ya que k es N y N es 8); en fila 1 y columna j=6, habrá un 7, ya que k se restó en 1; en fila 2 y columna j=0, habrá un 6; en fila 3 y columna 7, habrá un 5; en fila 4 y columna 3, hallamos un 4; fila 5 y columna 2, hay un 3; fila 6 y columna 4, hay un 2; y, finalmente, en la fila 7 y columna 1, habrá un 1.

17. Utilizando arreglos, escriba un programa que convierta un número de base N a base M y viceversa.

    Hay que tener una variable auxiliar, que nombraré aux con valor de 0, un arreglo de caracteres en donde será almacenado el número de base N, en mi caso, lo llamaré num con valores de 0 en todas las casillas, una variable llamada base10 con valor de 0, y también un arreglo de caracteres que contendrá el número convertido, que tendrá de nombre numM con valores de 0; además, los arreglos deben tener una longitud suficientemente grande como para que no se desborde.

    Antes de comenzar, se obtendrá la cantidad de cifras que haya en el arreglo mediante un bucle con contador k=0, mientras la casilla k del arreglo no se encuentre con el final de cadena, k aumentará en uno. El valor de k al final del bucle será la cantidad de cifras del número; declaramos una variable L que tendrá asignado el valor de k.

A continuación, se describirá el procedimiento para pasar de base N a base M:

El plan es pasar el número de base N a base 10 y, de ahí, pasarlo a base M. Si el arreglo tiene caracteres alfabéticos (mayor a base decimal), entonces se empezará convirtiendo esos caracteres a mayúsculas usando el método de un ejercicio anterior, el cual consiste en un bucle con un contador i=0, mientras la casilla i no se encuentre con el final de cadena, i aumentará en una uno. Dentro de este bucle, se hallará una condición que comprobará que la posición i del arreglo de la cadena sea mayor a 96 (en ASCII la letra 'a' es representada por 97, así que se inicia desde aquí), si esto es verdadero, entonces la posición i del arreglo de la cadena se le asignará la letra representada por el carácter de esa casilla i menos 32 ('A' es el 65 en ASCII y entre 'a' y 'A' hay una diferencia de 32). Si la condición no se cumple, entonces no se hace nada a esa posición del arreglo. Y el bucle seguirá hasta que se encuentre el fin de cadena; cuando esto ocurra, el bucle acabará.

Luego, ya teniendo el arreglo con las cifras alfabéticas en mayúsculas, iniciamos un bucle con dos contadores, i=L-1 y j=0, mientras i sea mayor o igual a 0, i se restará en uno al final de cada iteración y j aumentará en uno. Dentro de este bucle, habrá una condición que verificará que el valor en ASCII de la casilla i esté en un rango mayor a 47 y menor a 58, si esto es cierto, entonces la variable aux tendrá el resultado de la resta entre la casilla i en ASCII y 48; en la misma condición, la variable base10 tendrá el valor de la suma de sí mismo y el producto de aux con el número N elevado a la j. En caso de que la casilla no esté en ese rango de número, se comprobará con otra condición si el valor en ASCII de la casilla i se encuentra en un rango mayor de 63 y menor a 91. Si esto se cumple, entonces aux será el resultado de la resta entre la casilla i en ASCII y 55; luego, en la misma condición, base10 tendrá el valor de la suma de sí mismo y el producto de aux con el número N elevado a la j. Después de esto, se repetirá este procedimiento hasta que i sea menor a 0, es decir, hasta que i sea -1.

Ya teniendo el número N convertido a base 10, asignamos a aux el valor de 0 y comenzaremos otro bucle que hará la conversión de base 10 a base M. El bucle tendrá un contador i=0, mientras base10 sea mayor o igual a 0, i aumentará en uno al final de cada iteración; se empezará con que aux será el residuo de la división del número base10 entre el número M; luego, habrá una condición que comprobará que aux sea mayor a 0 pero menor a 10, si es verdadero, entonces en la casilla i del arreglo numM será asignado el valor de aux. Si no se cumple la condición, se pasa a que en la casilla i de numM será la suma de aux con 'A' y esto se restará con 10. Esto hará que, si aux es mayor o igual a 10, entonces se tendrá su representación en letra. Terminando la condición, la variable base10 tendrá el

valor del cociente de la división entre base10 y el número M. Después, se repiten estas instrucciones hasta que finalice el bucle.

Ahora, sin embargo, el orden de los números no es el correcto, y para solucionar esto, se invertirá la cadena del arreglo, pero antes se obtendrá la longitud de la cadena con el método mencionado en párrafos anteriores. Una vez teniendo la longitud de cadena L, se declarará un arreglo de caracteres de longitud L que llamaré baseM, ahí se tendrá el resultado final y en un bucle con contador i=0, mientras i sea menor a la longitud de cadena, i aumentará en uno al final de cada iteración; dentro del bucle, la casilla i de baseM será la casilla l-(i+1) de numM. Esta instrucción se repetirá hasta el fin del bucle. Después de que finalice el bucle, le añadimos el final de cadena en la casilla l.

18. Escriba un programa que realice la conversión de un número complejo en representación cartesiana $z = x + iy$, a una representación polar $z = re^{i\theta}$, y viceversa. Extienda su programa para que realice la suma y el producto de dos números complejos en cualquier representación.

    De cartesiano a polar, r es la raíz cuadrada de x al cuadrado sumado al valor de y elevado al cuadrado; el ángulo está dado por el resultado del arco tangente de la división del valor de y entre el valor de x (resultado en grados). Este ángulo será el exponente de e.

    Para polares a cartesianas, x es el producto de r por el coseno del ángulo; al similar sería con y, su valor es la multiplicación de r por el seno del mismo ángulo.

    Para la suma, hay que asegurarnos que ambos números complejos estén en su forma cartesiana, si no, habrá que convertirlos con el método anterior. Una vez teniéndolos, se suman los términos semejantes, es decir, x0 se suma con x1 & iy0 se suma con iy1. El resultado de esta agrupación será la suma de estos números complejos.

    La multiplicación de números complejos en representación cartesiana es posible; teniendo z0*z1 = (x0+iy0)*(x1+iy1) se obtiene x0*x1 + ix0*y1 + ix1*y0 + i^2y0*y1, como i=$\sqrt{-1}$, entonces i^2= -1; por lo tanto, z0*z1 = x0*x1 – y0*y1 + i(x0*y1 + x1*y0) = (x0*x1 – y0*y1) + i(x0*y1 + x1*y0). Esto sería la multiplicación de complejos.

    También es posible el producto de números complejos en su representación polar. El producto se realiza con la multiplicación de los radios r (r0*r1) y esto es multiplicado por e elevado a la suma de los ángulos de los números complejos

(e^i(ángulo0+ángulo1). Esto es el producto de los números complejos. La suma en forma polar no es posible.

19. Complete el código para multiplicar las siguientes matrices cuadradas de 3x3 y calcule el contenido de las celdas de la matriz resultante $C = AB$:

$$A = \begin{bmatrix} 1 & 7 & 5 \\ 8 & 4 & 0 \\ 3 & 6 & 3 \end{bmatrix}, B = \begin{bmatrix} 4 & 6 & 7 \\ 3 & 1 & 2 \\ 8 & 5 & 0 \end{bmatrix}$$

$$C_{ij} = \sum_{k=0}^{N-1} A_{ik} B_{kj}$$

```
int N=3;
int matA[N][N], matB[N][N], matC[N][N];
int i=0, j=0, k=0;

for (i=0; i<N; i++)
   {
      for (j=0; j<N; j++)
     {
       matC[i][j]= 0;
       for (k=0; k<N; k++)
          matC[i][j]+= matA[i][k]*matB[k][j];
     }
   }
```

Se tienen tres arreglos matrices A, B y C; los primeros dos ya están inicializados y el tercero se inicializará con un 0 en la posición (i, j) de la matriz C dentro del segundo bucle. En el tercer bucle se realiza la sumatoria de la casilla (i, j) de la matriz C con el producto de cada casilla de la fila (i, k) de la matriz A por cada casilla de la columna (k, j) de la matriz B, en donde únicamente se multiplicarán las casillas que tengan valor k iguales; esto es, por ejemplo, si se tiene (i=1, k=2) en A y (k=2. j=1) en B, entonces el contenido de estas dos casillas se multiplicarán entre sí. Con la variable k se asegura que cada casilla de la fila i de A se multiplique por cada casilla de la columna j de B, luego, cuando j aumenta en uno, se hace el producto de toda la fila i de A con toda la otra columna j de B y así hasta que se deje de cumplir la condición del bucle de j, para que luego pase a que i aumente en uno y se repita el proceso.

El contenido de la matriz C es:

Cuando i=0: (0, j=0) = (1*4 + 7*3 + 5*8=) 65; (0, j=1) = (1*6 + 7*1 + 5*5=) 38; (0, j=0) = (1*7 + 7*2 + 3*0=) 21.

Cuando i=1: (1, j=0) = (8*4 + 4*3 + 0*8=) 44; (1, j=0) = (8*6 + 4*1 + 0*5=) 52; (1, j=0) = (8*7 + 4*2 + 0*0=) 64.

Cuando i=2: (2, j=0) = (3*4 + 6*3 + 3*8=) 54; (2, j=0) = (3*6 + 6*1 + 3*5=) 39; (2, j=0) = (3*7 + 6*2 + 3*0=) 33.

Puesto que la condición del bucle marca que mientras i sea menor a N, el cual es 3, entonces cuando i alcance ese número, el bucle finalizará.

20. Escriba el contenido del siguiente arreglo cúbico:

```
int N=3;
int volumen[N][N][N];
int i=0, j=0, k=0;

for (i=0; i<N; i++)
   for (j=0; j<N; j++)
      for (k=0; k<N; k++)
         volumen[i][j][k]= i*j*k;
```

Considerando que i es la altura de este arreglo cúbico, puede dividirse en N partes, tomando de referencia a i, para describir las matrices de mejor manera.

Cuando i=0, todas las casillas de esta sección son 0's, ya que i*j*k, donde i es 0, da 0.

Cuando i=1, toda la primera fila y columna es 0, ya que, en el primer caso, tenemos j=0; y en el segundo caso, al inicio de cada fila k es 0. Sin embargo, para j>0 y k>0, se tiene que en el valor de cada casilla es el producto de la fila y la columna, teniendo así: (1,1) =1, (1,2) =2, (2,1) =2, (2,2) =4. La multiplicación con i no se menciona ya que no afecta al resultado.

Cuando i=2, tenemos que, igualmente, toda la primera fila y columna es 0 por las mismas razones que lo anterior. En cambio, en la multiplicación de casillas, tenemos que en donde había un 1, ahora hay un 2 debido a la multiplicación por i=2; donde había un 2, ahora es un 4; y en lugar del 4, ahora es 8. Esto quiere decir que los valores de las casillas de la matriz anterior, en esta matriz, tendrán un valor doble.