Relación de ejercicios tema 2: punteros y gestión dinámica de memoria

Metodología de la programación, 2015-2016

Contenido:

1 Problemas básicos

1. Describid la salida de los siguientes programas:

   (a)

```
#include <iostream>
using namespace std;

int main (){
  int a = 5, *p;

  a = *p * a;
  if (a == *p)
    cout << "a es igual a *p" << endl;
  else
    cout << "a es diferente a *p" << endl;
}
```

   (b)

```
#include <iostream>
using namespace std;

int main (){
  int a = 5, *p;
```

```
  *p = *p * a;
  if (a == *p)
    cout << "a es igual a *p" << endl;
  else
    cout << "a es diferente a *p" << endl;
}
```

(c)

```
#include <iostream>
using namespace std;

int main (){
  int a = 5, *p = &a;

  *p = *p * a;
  if (a == *p)
    cout << "a es igual a *p" << endl;
  else
    cout << "a es diferente a *p" << endl;
}
```

(d)

```
#include <iostream>
using namespace std;

int main (){
  int a = 5, *p = &a, **p2 = &p;

  **p2 = *p + (**p2 / a);
  *p = a+1;
  a = **p2 / 2;
  cout << "a es igual a: " << a << endl;
}
```

2. Dado un vector de 10 elementos, haz un bucle que busque el máximo y el mínimo (sin usar el operador `[]`). Al acabar el bucle tendremos un puntero apuntando a cada uno de ellos.

3. Dado un vector de 10 elementos, escribe un programa que invierta el vector (sin usar el operador `[]`).

4. Declare una variable `v` como un vector de 1000 enteros. Escriba un trozo de código que recorra el vector y modifique todos los enteros negativos cambiándolos de signo. No

se permite usar el operador `[]`, es decir, el recorrido se efectuará usando aritmética de punteros y el bucle se controlará mediante un contador entero. Nota: Para inicializar aleatoriamente el vector con valores enteros entre -50 y 50, por ejemplo, puede emplearse el siguiente fragmento de código:

```
#include <cstdlib>
#include <ctime>
...
// Queremos valores -50<=n<=50
const int MY_MAX_RAND = 50;
time_t t;
...
// Inicializa el generador con el reloj
// del sistema
srand ((int) time(&t));
...
for (int i=0; i<1000; i++)
    v[i] =  (int)(1.0*(MY_MAX_RAND+MY_MAX_RAND+1)*rand()/
                            (RAND_MAX+1.0) - MY_MAX_RAND );
```

En general, para generar números aleatorios entre inf y sup haremos:

`(int) (1.0*(sup-inf+1)*rand()/(RAND_MAX+1.0)) + inf;`

Se puede obtener más información sobre srand(), rand() y time() en la siguiente dirección: `http://www.cplusplus.com`.

5. Modifique el código del problema anterior para controlar el final del bucle con un puntero a la posición siguiente a la última.

6. Supongamos las siguientes declaraciones:

```
const int Max = 100;
float v1 [Max] = {2, 3, 8, 22, 44, 88, 99, 100, 101, 255, 665};
float v2 [Max] = {1, 3, 4, 5, 6, 25, 87, 89, 99, 100, 500, 1000};
float res [2*Max];
int tam_v1=11, tam_v2=12; // 0 <= tam_v1, tam_v2 < Max
int tam_res = tam_v1+tam_v2; // 0 <= tam_res < 2*Max
```

Nota: Observad que `v1` y `v2` almacenan valores ordenados de menor a mayor.

Escribir un trozo de código para mezclar, de manera ordenada, los valores de `v1` y `v2` en el vector res. No se debe usar el operador `[]`, es decir, se debe resolver usando aritmética de punteros.

7. Consideremos un vector v de números reales de tamaño `n`. Supongamos que se desea dividir el vector en dos secciones: la primera contendrá a todos los elementos menores o iguales al primero (`v[0]`) y la otra, los mayores. Para ello, proponemos un algoritmo que consiste en:

   - Colocamos un puntero al principio del vector y lo adelantamos mientras el elemento apuntado sea menor o igual que el primero.

- Colocamos un puntero al final del vector y lo atrasamos mientras el elemento apuntado sea mayor que el primero.
- Si los punteros no se han cruzado, es que se han encontrado dos elementos *mal colocados*. Los intercambiamos y volvemos a empezar.

Este algoritmo acabará cuando los dos punteros se crucen, habiendo quedado todos los elementos ordenados según el criterio inicial.

Escriba un trozo de código que declare una constante (`n`) con valor 20 y un vector de reales con ese tamaño, lo rellene con números aleatorios entre 0 y 100 y lo reorganice usando el algoritmo antes descrito.

8. Las cadenas de caracteres representan un ejemplo clásico en el uso de punteros. La cadena almacena un número indeterminado de caracteres (para los ejercicios bastará un valor siempre menor que 100) delimitados al final por el carácter nulo (`'\0'`).

   Escriba un trozo de código que localice la posición del primer carácter espacio (`' '`) en una cadena de caracteres *clásica*.

   La cadena debe recorrerse usando aritmética de punteros y sin usar ningún entero. El programa debe indicar su posición (0: primer carácter, 1: segundo carácter, etc.).

   Nota: Usar la función `getline()` para la lectura de la cadena (Cuidado: usar el método público de `iostream` sobre `cin`, o sea `cin.getline()`). Ver `http://www.cplusplus.com/reference/iostream/istream/getline/`

9. Consideremos una cadena de caracteres estilo C. Escriba un trozo de código que lea una cadena y la imprima pero saltándose la primera palabra, evitando escribirla carácter a carácter. Considere que puede haber una o más palabras, y si hay más de una palabra, están separadas por espacios en blanco.

10. Implementa una función que reciba un puntero a entero y que:

    (a) Ponga a cero el dato apuntado.

    (b) Ponga a cero el puntero.

```
int a=6;
int *q;
q = &a;
HacerCero(q);
cout << a << q; // Debería salir 0 0
```

11. Indica si se modifica o no, `v[3]` en el siguiente código:

```
void duplica(int v[], int p) {
    v[p] *= 2;
}
int main(){
    int v[6] = {5,4,3,7,6,8};
    duplica(v,3);
    cout << v[3] << endl;
}
```

12. Indica qué ocurre en las siguientes situaciones:

(a)
```
int *p;
const int a=2;
p = &a;
```

(b)
```
const int *p;
int a;
p = &a;
*p = 7;
a = 8;
```

(c)
```
int *v1;
int * const v2;
const int * v3;
const int * const v4;
```

```
v1 = v2;
v1 = v3;
v1 = v4;
```

```
v2 = v1;
v2 = v3;
v2 = v4;
```

```
v3 = v1;
v3 = v2;
v3 = v4;
```

```
v4 = v1;
v4 = v2;
v4 = v3;
```

```
*v1=*v2;
*v1=*v3;
*v1=*v4;
```

```
*v2=*v1;
*v2=*v3;
*v2=*v4;
```

```
*v3=*v1;
*v3=*v2;
*v3=*v4;
```

```
*v4=*v1;
*v4=*v2;
*v4=*v3;
```

13. Describid la salida del siguiente programa:

```
#include <iostream>
using namespace std;

int main (){
   int *p1, *p2;

   p1 = new int;
   *p1 = 42;
   p2 = p1;
   cout << *p1 << " y " << *p2 << endl;

   *p2 = 53;
   cout << *p1 << " y " << *p2 << endl;

   p1 = new int;
   *p1 = 88;
   cout << *p1 << " y " << *p2 << endl;
}
```

2 Punteros y funciones

14. Implemente las siguientes funciones sobre cadenas de caracteres estilo C:

(a) Función `longitud_cadena`. Devuelve un entero con la longitud (número de caracteres sin contar el nulo) de la cadena.

(b) Función `comparar_cadenas`. Compara dos cadenas. Devuelve un valor negativo si la primera es más *pequeña*, positivo si es más *grande* y cero si son *iguales*.

(c) Función `copiar_cadena`. Copia una cadena de caracteres en otra. El resultado es el primer argumento.

(d) Función `encadenar_cadena`. Añade una cadena de caracteres al final de otra. El resultado es el primer argumento.

Se supone que hay suficiente memoria en las cadenas de destino y no es necesario pasar el tamaño de las cadenas (recordad que el carácter nulo delimita el final de la cadena).

15. Escriba una función a la que le damos una cadena de caracteres, una posición de inicio `p` y una longitud `l`. Como resultado, devuelve una subcadena que comienza en `p` y que tiene longitud `l`. **Nota**: Si la longitud es demasiado grande (se sale de la cadena original), se devolverá una cadena de menor tamaño. No se debe usar el operador `[]`, es decir, se debe resolver mediante aritmética de punteros.

16. Se desea una función que reciba un vector de números enteros junto con su longitud y que devuelva un puntero al elemento mayor.

    Escriba dos versiones:

    (a) Devuelve el resultado como resultado de la función (`return`).

    (b) Devuelve el resultado a través de un parámetro (función `void`).

    Considere la siguiente declaración:

    ```
    int vector [100];
    int usados; // 0 <= usados < 100
    ```

    Haga uso de la primera función para mostrar en la salida estándar:

    (a) El elemento mayor del vector.

    (b) El elemento mayor de la primera mitad.

    (c) El elemento mayor de la segunda mitad.

17. Escriba una función que reciba como entrada un vector de números junto con su longitud y que nos devuelva un vector de punteros a los elementos del vector de entrada de forma que los elementos apuntados por dicho vector de punteros estén ordenados (véase la siguiente figura). Note que el vector de punteros debe ser un parámetro de la función, y estar reservado previamente a la llamada con un tamaño, al menos, igual al del vector. Una vez escrita la función, considere la siguiente declaración:

    ```
    int vec [1000];
    int *ptr [1000];
    ```

y escriba un trozo de código que, haciendo uso de la función, permita:

(a) Ordenando punteros, mostrar los elementos del vector, ordenados.

(b) Ordenando punteros, mostrar los elementos de la segunda mitad del vector, ordenados.

sin modificar el vector de datos `vec`.

3 Punteros y clases

18. Represente gráficamente la disposición en memoria de las variables del programa mostrado a continuación, e indique lo que escribe la última sentencia de salida.

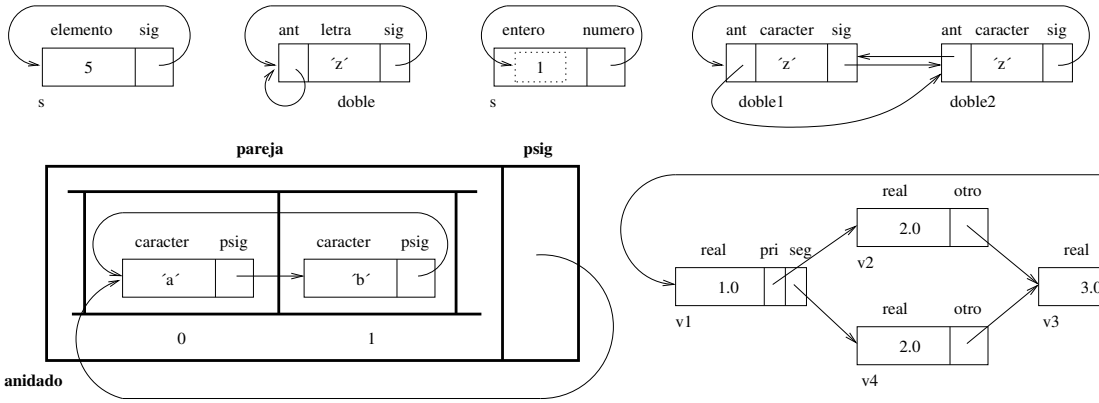
```
#include <iostream>
using namespace std ;

class Celda
{
  public:
    int d ;
    Celda *p1 , *p2 , *p3 ;
};

int main ( int argc , char * argv [ ] )
{
    Celda a, b, c, d;

    a.d = b.d = c.d = d.d = 0 ;

    a.p1 = &c;
    c.p3 = &d;
    a.p2 = a.p1->p3;
    d.p1 = &b;
    a.p3 = c.p3->p1;
    a.p3->p2 = a.p1;
    a.p1->p1 = &a;
    a.p1->p3->p1->p2->p2 = c.p3->p1;
    c.p1->p3->p1 = &b;
    (*((*(c.p3->p1)).p2->p3)).p3 = a.p1->p3;
    d.p2 = b.p2;
    (*(a.p3->p1)).p2->p2->p3 = (*(a.p3->p2)).p3->p1->p2 ;

    a.p1->p2->p2->p1->d = 5;
    d.p1->p3->p1->p2->p1->p1->d = 7 ;
    (*(d.p1->p3)).p3->d = 9 ;
    c.p1->p2->p3->d = a.p1->p2->d - 2 ;
    (*(c.p2->p1)).p2->d = 10 ;

    cout << "a="<<a.d<<" b="<<b.d<<" c="<<c.d<<" d="<<d.d<<endl ;
}
```

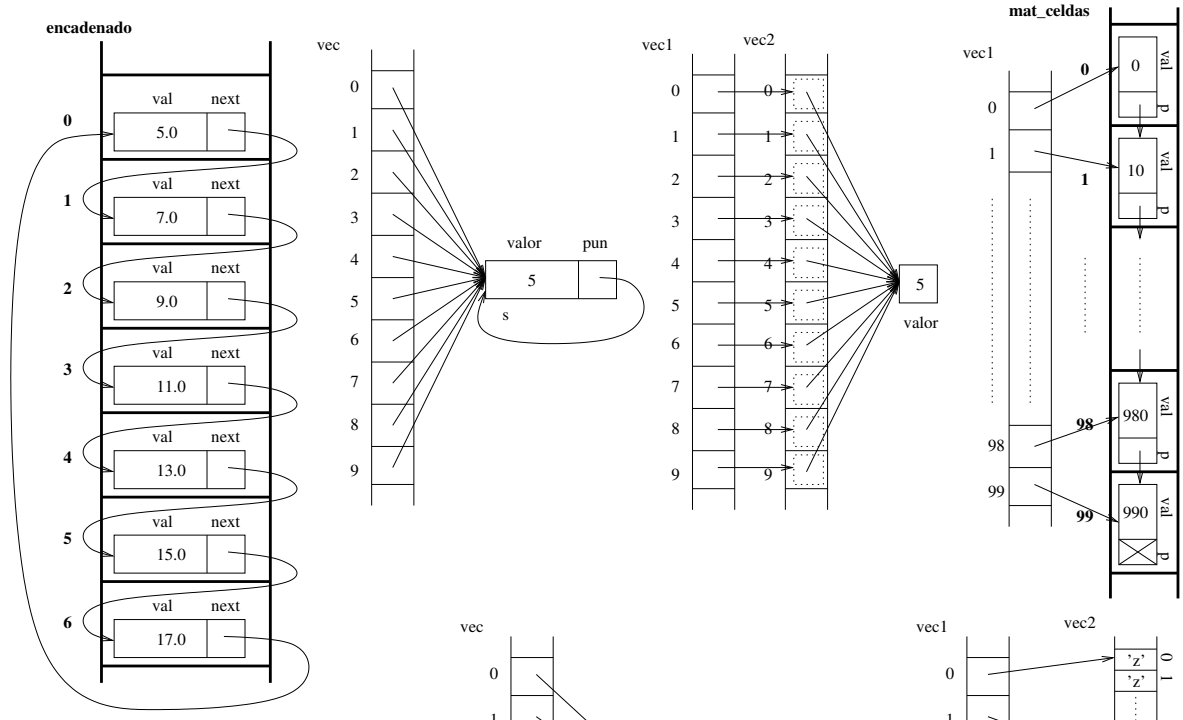
19. Represente gráficamente la disposición en memoria de las variables del programa mostrado a continuación, e indique lo que escribe la última sentencia de salida. Tenga en cuenta que el operador `->` tiene más prioridad que el operador `*`.

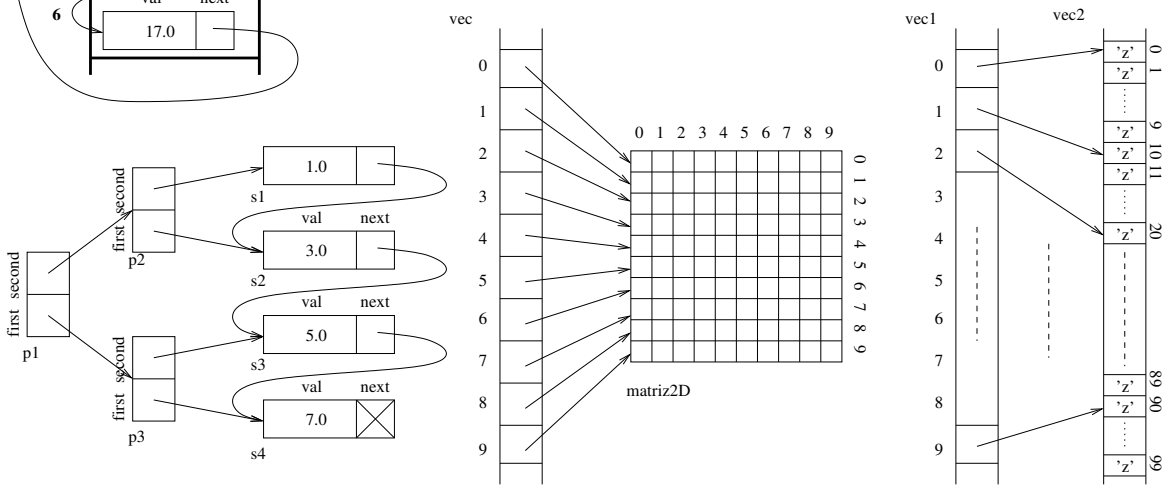
```
#include <iostream>
using namespace std ;

class SB; // declaración adelantada
class SC; // declaración adelantada
class SD; // declaración adelantada

class SA {
  public:
    int dat ;
    SB *p1 ;
} ;

class SB {
  public:
    int dat ;
    SA *p1 ;
    SC *p2 ;
} ;

class SC {
  public:
    SA *p1 ;
    SB *p2 ;
    SD *p3 ;
} ;

class SD {
  public:
    int *p1 ;
    SB *p2 ;
} ;

int main ( int argc , char * argv [ ] )
{
    SA a ;
    SB b ;
    SC c ;
    SD d ;
    int dat ;

    a.dat = b.dat = dat = 0 ;

    a.p1 = &b ;
    b.p1 = &a ;
    b.p2 = &c ;
    c.p1 = b.p1 ;
    c.p2 = &(*(a.p1)) ;
    c.p3 = &d;
    d.p1 = &dat;
    d.p2 = &(*(c.p1)->p1);
```

```
    *(d.p1) = 9;
    (*(b.p2)->p1).dat = 1;
    (*((b.p2)->p3->p2)->p1).dat = 7 ;
    *((*((*(c.p3->p2)).p2->p3)).p1) = (*(b.p2)).p1->dat + 5 ;
    cout << "a.dat=" << a.dat << " b.dat=" << b . dat << " dat=" << dat << endl ;
}
```

20. Considere la siguiente figura. Se presentan gráficamente un conjunto de estructuras de datos. Se puede observar que las matrices se representan indicando los índices y las clases o estructuras indicando los nombres de los datos miembro o campos. Escriba los trozos de código que corresponden a su creación.


elemento sig
5
s
ant letra sig
´z´
doble
entero numero
1
s
ant caracter sig
´z´
doble1
ant caracter sig
´z´
doble2
pareja
psig
caracter psig
´a´
caracter psig
´b´
0
1
anidado
real otro
2.0
real pri seg
1.0
v1
v2
real otro
3.0
v3
real otro
2.0
v4


encadenado
val next
5.0
7.0
9.0
11.0
13.0
15.0
17.0
0
1
2
3
4
5
6
vec
valor pun
5
s
vec1
vec2
valor
mat_celdas
vec1
0
10
980
990
val p
0
1
98
99


first second
p1
p2
p3
1.0
s1
val next
3.0
s2
5.0
s3
7.0
s4
vec
matriz2D
vec1
vec2
'z'
0 1 9 10 11 20 89 90 99

21. Dadas las siguientes declaraciones

```
class Electrica {
  public:
   char corriente[30];
   int voltios;
};

Electrica *p = new Electrica(), *q = new Electrica();
```

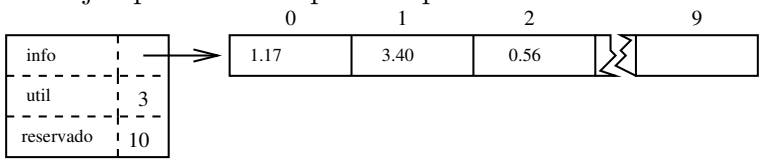
se pide averiguar qué hacen cada una de las siguientes sentencias. ¿Hay alguna inválida?

```
a.  strcpy(p->corriente,"ALTERNA");   e.  strcpy(p->corriente,"ALTA");
b.  p->voltios = q->voltios;          f.  p->corriente = q->voltios;
c.  *p = *q;                          g.  p = 54;
d.  p = q;                            h.  *q = p;
```

22. Supongamos la siguiente definición de tipo de dato:

```
class VectorSD {
  private:
   double *info;
   int util;
   int reservado;
};
```

donde `info` es un puntero que mantiene la dirección de una secuencia de reales, `util` indica el número de componentes de la secuencia y `reservado` indica el número de posiciones reservadas de la memoria dinámica para almacenar la secuencia de reales. La siguiente figura muestra un ejemplo de este tipo de representación.



a) construid un método de dicha clase que inicialice una variable de tipo `VectorSD` reservando **n** casillas de memoria dinámica y ponga el número de componentes usadas a 0.

b) construid un método que añada un elemento en una variable `VectorSD`. Considerar el caso de que la inclusión del nuevo valor sobrepase la reserva de memoria. En este caso, realojar el vector reservando el doble de posiciones.

c) construid un método que realice una copia de una variable `VectorSD` en otra variable del mismo tipo. La copia debe reservar memoria para almacenar sólo las componentes usadas del vector.

d) construir un método que libere la memoria reservada por una variable de tipo `VectorSD`.

e) implementad un método que devuelva el elemento que ocupa una determinada posición.

f) implementad una función que indique el número de elementos contenidos en el vector.

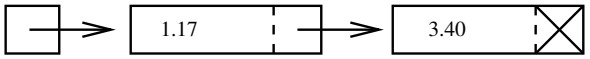
g) implementad un constructor para la clase.

23. Dadas las siguientes definiciones de tipos de dato:

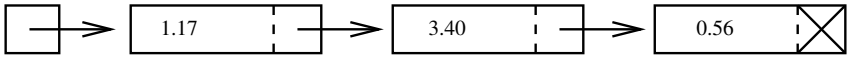
```
class Celda{
  private:
   double info;
   Celda *sig;
  public:
    ............
};

class Lista{
  private:
    Celda * contenido;
  .......
};
```

donde `info` es una variable `double` y `sig` es un puntero que apunta a una variable `Celda`.

a) implementad un método que permita añadir al final de una secuencia de celdas enlazadas una nueva celda. Por ejemplo, dada la siguiente situación:



el resultado de añadir el valor `0.56` al objeto correspondiente sería:



NOTA: El puntero de la última celda contiene un 0.

b) construid un método que permita eliminar la última celda de un objeto de la clase.

c) construid un método que elimine y libere toda la información contenida en un objeto.

d) construid un método que muestre en la salida estándar el contenido completo de un objeto de la clase.

e) construid un método que inserte una nueva celda detrás de otra celda concreta (cuya dirección de memoria se pasa como parámetro a la función).

f) ¿podría utilizarse el método anterior para insertar una celda al principio de la lista? Si no es así, haced los cambios necesarios para que sea posible.

g) implementad un método que inserte la nueva celda delante de la celda indicada.

h) construid un método que, dada la dirección de memoria de una celda, la elimine de la lista.

i) ¿sería posible utilizar el método anterior para eliminar la primera de las celdas de la estructura de celdas enlazadas? Si no es así, haced los cambios necesarios para que sea posible.

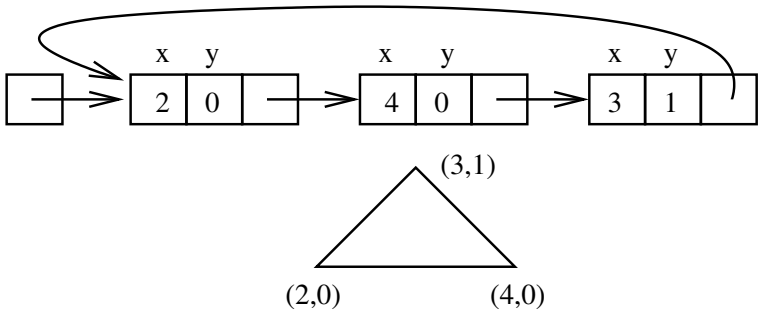
j) implementad el método de ordenación de la burbuja para ordenar los elementos de la lista.

k) suponiendo que partimos un objeto de la clase en que las celdas enlazadas están ordenadas de forma ascendente, implementad un método que inserte ordenadamente una nueva celda.

24. Se desea desarrollar una clase que permita representar de forma general diversas figuras poligonales. Cada figura poligonal se puede representar como un conjunto de puntos en el plano unidos por segmentos de rectas entre cada dos puntos adyacentes. Por esta razón se propone la siguiente representación:

```
class Punto2D{
  private:
   double x;
   double y;
  public:
    ........
};

class Nodo{
  private:
   Punto2D punto;
   Nodo *sigPunto;
  public:
    .........
};

// Clase Poligono
class Poligono{
   private:
     Nodo* figura;
    ...........
}
```

Dada esta definición, un polígono se representa como una secuencia circular ordenada de nodos enlazados, por ejemplo, el triángulo de puntos (2,0),(4,0) y (3,1) se representa de la siguiente forma:



Teniendo en cuenta está representación, responder a las siguientes cuestiones:

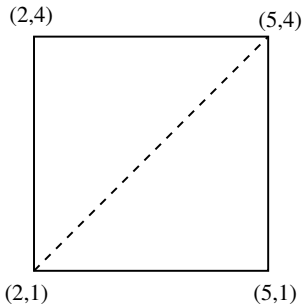
a) construid un método que determine el número de lados que contiene la figura almacenada en una variable de tipo `Poligono`.

b) suponiendo que existe un método llamado `pintaRecta (Punto2D p1, Punto2D p2)` que pinta una recta entre los 2 puntos que se le pasan como argumento, construid un método que permita pintar la figura que representa una determinada variable `Poligono`.

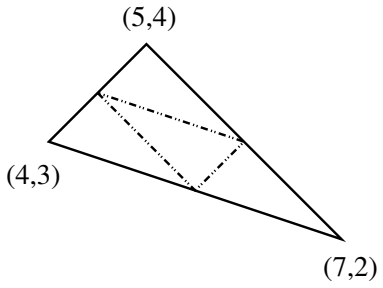
c) implementad un método que permita crear en una variable de tipo `Poligono` un triángulo a partir de los tres puntos que lo definen.

d) desarrollad un método que permita liberar la memoria reservada por una variable `Poligono`.

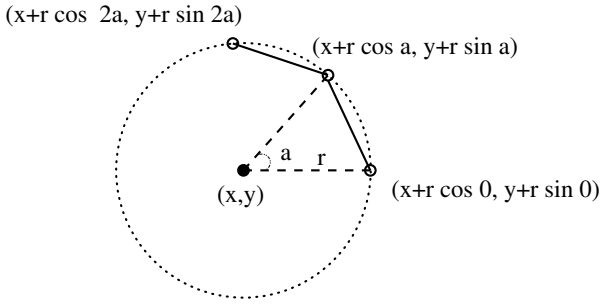
e) sabiendo que una variable `Poligono` almacena un cuadrado, implementar un método que devuelva los dos triángulos que resultan de unir mediante una recta la esquina inferior izquierda del cuadrado con su esquina superior derecha.



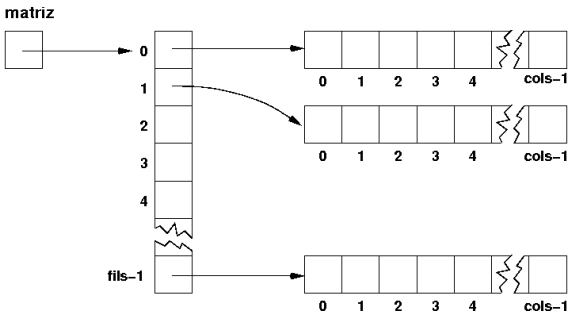
f) construir una función que a partir de una variable `Poligono` que representa un triángulo devuelva el triángulo formado por los puntos medios de las rectas del triángulo original.



g) desarrollad un método que permita construir un polígono regular de **n** lados inscrito en una circunferencia de radio **r** y centro **(x,y)**.



25. Supongamos que para definir matrices bidimensionales dinámicas usamos una estructura como la que aparece en la siguiente figura que denominaremos **Matriz2D_1**:



a) define cómo sería la clase para representar una matriz con esta estructura.

b) añade el constructor de la clase para crear una matriz con un determinado número de filas y de columnas.

c) construid un método que escriba todas las componentes de una matriz bidimensional dinámica como la que se ha definido anteriormente.

d) construid un método que dada una matriz de este tipo cree una copia.

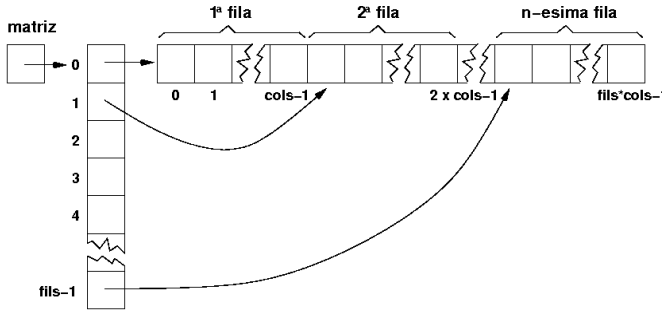
e) implementad un método que extraiga una submatriz de una matriz bidimensional `Matriz2D_1`. Como argumento de la función se introduce desde qué fila y columna y hasta qué fila y columna se debe realizar la copia de la matriz original.

f) desarrollad un método que elimine una fila de una matriz bidimensional `Matriz2D_1`. Obviamente, la eliminación de una fila implica el desplazamiento hacia arriba del resto de las filas que se encuentran por debajo de la fila eliminada.

g) realizad un método como el anterior, pero que en vez de eliminar una fila, elimine una columna.

26. Supongamos que ahora decidimos utilizar una forma diferente para representar las matrices bidimensionales dinámicas a la que se propone en el ejercicio anterior. En este caso,

usaremos una estructura semejante a la que aparece en la siguiente figura que denominaremos **Matriz2D_2**:



a) realizad los apartados a. . .f usando esta nueva representación de matrices bidimensionales dinámicas.

b) construid un método que dada una matriz bidimensional dinámica `Matriz2D_1` realice una copia de la misma en una matriz bidimensional dinámica `Matriz2D_2`.

c) desarrollad un método que realice el paso inverso al propuesto en el ejercicio anterior.