```
int i
      #include<iostream>

    Return 0

While(i <= j){
        }

if(x > 4){
     }

 10100100100

00001100100

  11101010010

 0101

0
1
1
1
0
```

# Arreglos

Prof. Nicolás Hidalgo
nicolas.hidalgoc@mail.udp.cl

# Arreglos

- Tipo estructurado de dato que almacena una colección de datos del mismo tipo

- Estructura de datos ampliamente usada por programadores

- Es una forma simple de agrupar componentes de un mismo tipo , ordenados y referenciados por un índice

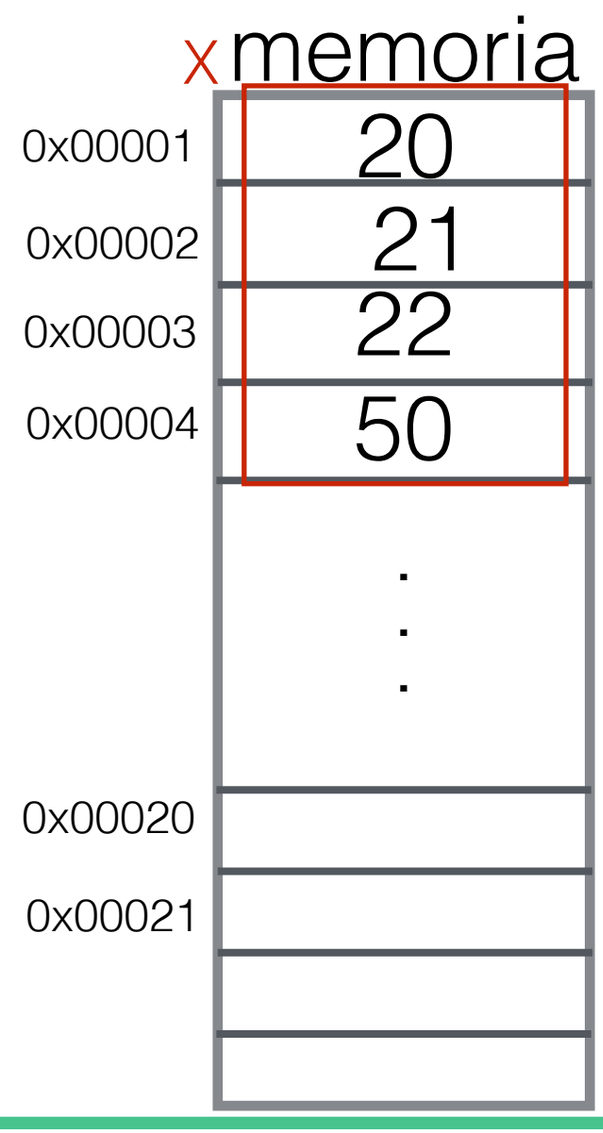
- Ojo, los arreglos en C/C++ se guardan en posiciones contiguas en memoria

tipo nombre [dimensión-tamaño];

int arr[4];

Tamaño n ->  0… n-1

for(int i=0 ; I < 4 ; i++)
Cout << arr[i];




Int x[n+1];  Indice ->[0, n]
Cout << x[indice];

Int x[4]; //tamaño

Cout << x[0];  //indices
Cout << x[1];
Cout << x[2];
Cout << x[3];

x[3] = 50;

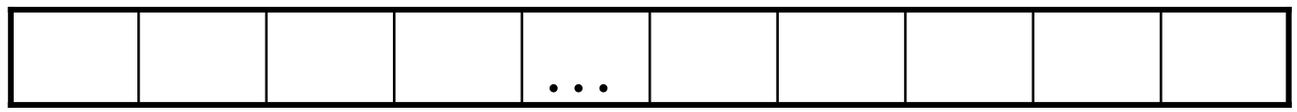
# Arreglos

- Un arreglo es una estructura multidimensional, sin embargo, por lo general se trabaja con una o dos dimensiones
  - **Uni-dimensional** (vector): se utiliza 1 índice para acceder a los datos
  - **Bi-dimensional** (matriz): se necesitan 2 índices para acceder a un dato (fila y columna)
- En general, si tenemos una estructura basada en arreglos de n dimensiones, se requerirán **n** índices para acceder a los datos

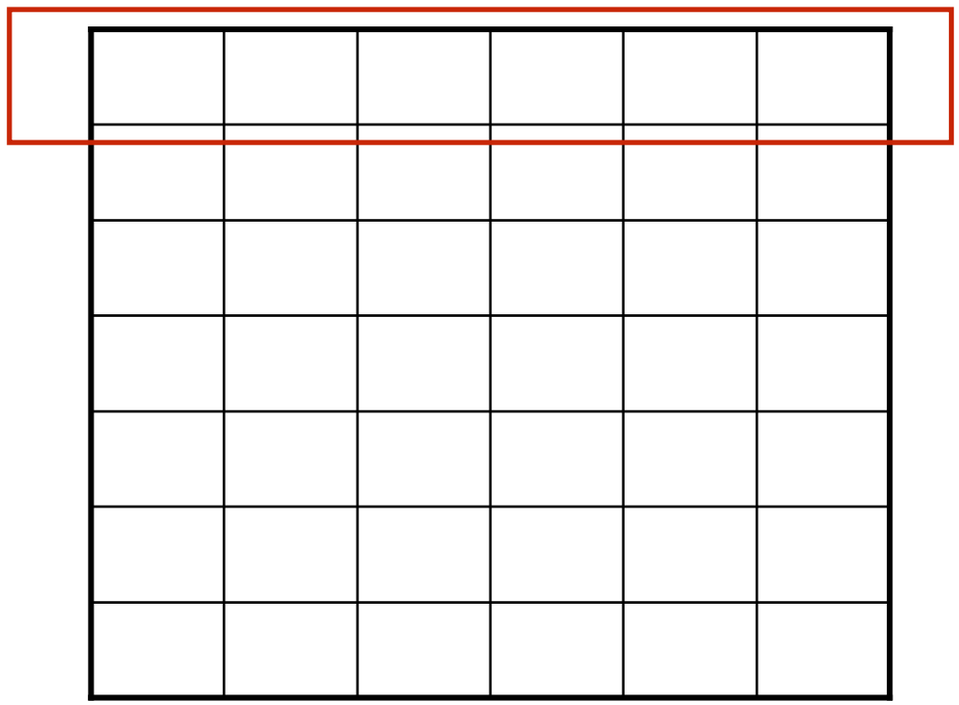
# Arreglos

**Unidimensional**

**int a[20];**



Int a [7][6];

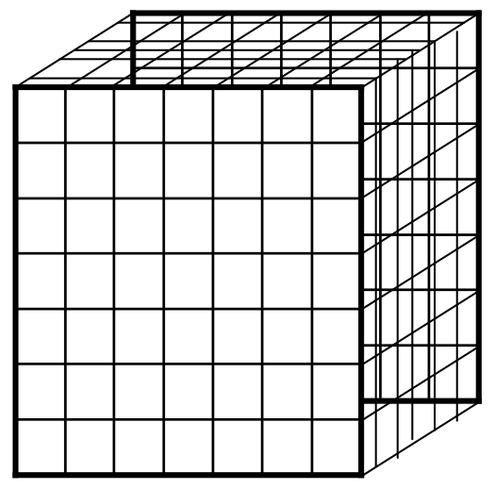


**Bidimensional**

**Int a[6][7];**

**Tridimensional**

**float a[7][7][4];**



# Arreglos

- Ojo, los indices de los arreglos comienzan en cero, es decir el primer elemento, es el elemento cero

- Los arreglos, al igual que las variables, tienen un tipo de dato especifico, un nombre, la dimensionalidad y un tamaño.

- `tipo_dato nombre_arreglo [tamano];`
  - **`tipo:`** tipo de elementos, todos son del mismo tipo
  - **`Nombre:`** nombre de la variable tipo arreglo
  - **`[]:`** indica dimensionalidad
  - t**`amano:`** cantidad de elementos del arreglo

# Arreglos

- Para acceder a un elemento del arreglo es necesario utilizar el **índice**

- El índice describe la posición en el arreglo que deseamos obtener

- En C/C++ el primer elemento es referenciado por el índice cero!

**int V[5];**

| **V →** | 13 | 34 | 7 | 61 | 1 |
|---|---|---|---|---|---|
| Posición | 0 | 1 | 2 | 3 | 4 |

**V[2]** contiene el **7**

**V[4]** contiene el **1**

**V[0]** contiene el **13**

# Algunos arreglos…

Indice 0 1 2 3 4 5

- `int numero[] = {1,2,3,4,5, 66}; // tamaño 6`
- `char vocales [5] = {a, e, i, o, u}; //tamaño 5`
- `char nombres[][6] = {“Pedro”, “Juan”, “Diego”}; //3 filas y columnas de máximo 6 caracteres`
- `int coordenadas[2][2] = {{0,0}, {1,1}};`

# Arreglos

- Se podrá hacer lo siguiente para copiar un arreglo a otro?

```
int a[10], b[10];
```

X `a = b;`

Para copiar:

```
for(int I = 0 ; I < 10 ; I++)
    a[i] = b[i];
```

# Arreglos

```
test.cpp:28:3: error: array type 'int [10]' is not
assignable
a = b;
```

- Debemos copiar cada elemento:

```
for(int i = 0; i < 10 ; i++)

    a[i]=b[i];
```

# Arreglos

- **Ejercicio 1**: realizar un programa en C++ en el que se ingresen 10 enteros por parte del usuario, y luego muestre la suma de ellos, su promedio, el mínimo y el máximo obtenidos desde un arreglo.

Indice:

# Arreglos

- **Ejercicio 1.1**: realizar un programa en C++ en el que se ingresen 10 enteros por parte del usuario, y luego muestre la suma de ellos, su promedio, el mínimo y el máximo. Resolver utilizando arreglos y dejar **mínimo en primera casilla** del arreglo y **máximo en la última casilla** del arreglo.

Indice:

# Arreglos

- **Ejercicio 2**: realizar un programa en C++ en el que se ingresen 10 enteros por parte del usuario, y luego muestre la suma de ellos, su promedio, el mínimo y el máximo. Resolver utilizando arreglos y funciones.
  - int suma(int array[]) { }
  - int min(int array[]) { }
  - int max(int array[]) { }

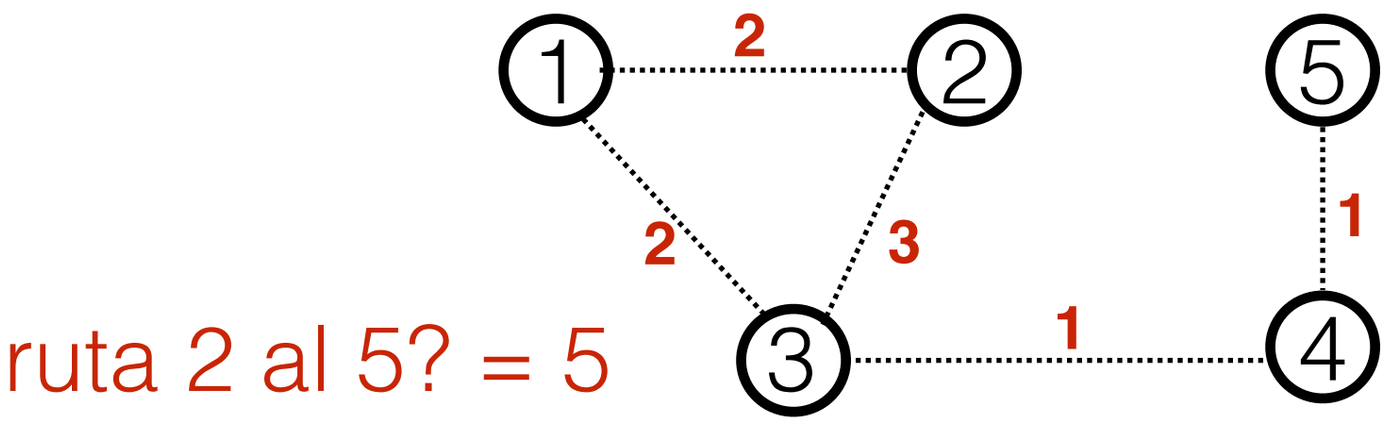
# Arreglos

- **Ejercicio 3**: realizar un programa en C++ en el que se ingresen 10 enteros por parte del usuario, y luego muestre la suma de ellos, su promedio, el mínimo y el máximo. Resolver utilizando arreglos, funciones y punteros.
  - int suma(int *array, int size) { }
  - int min(int *array, int size) { }
  - int max(int *array, int size) { }

# Arreglos Bidimensionales

- **Ejercicio 6**: Escriba un programa que permita ingresar los costos de ruteo de un grafo de red de tamaño N. El programa debe solicitar ingresar al usuario los costos de comunicación representados en la estructura gráfica del grafo.

- Ejemplo:




| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 2 | 2 | 999 | 999 |
| 2 | 2 | 0 | 3 | 999 | 999 |
| 3 | 2 | 3 | 0 | 1 | 999 |
| 4 | 999 | 999 | 1 | 0 | 1 |
| 5 | 999 | 999 | 999 | 1 | 0 |

# Arreglos Bidimensionales

- **Ejercicio 7:** Haciendo uso del programa anteriormente descrito, modifique para que el programa solicite un punto de inicio y un destino y entregue un camino a seguir para alcanzar el destino en caso de ser posible.

# Arreglos y punteros

- **Ejercicio 5**: Escriba un programa que solicite el ingreso de un Tweet y cuente el número de vocales presentes en el texto. Resuelva haciendo uso de funciones y punteros.

  - int cuentaVocales(char *s) { }

# String y Arreglos

- ¿Cómo podemos  copiar un string a un arreglo de caracteres?

```
int main()
{
    // assigning value to string s
    string s = "StringTest";

    int n = s.length();

    // declaring character array
    char char_array[n + 1];

    // copying the contents of the
    // string to char array
    strcpy(char_array, s.c_str());

    for (int i = 0; i < n; i++)
        cout << char_array[i];

    return 0;
}
```

```
int main()
{
    // assigning value to string s
    string s = "StringTest";

    int n = s.length();

    // declaring character array
    char char_array[n + 1];

    for (int i = 0; i < n; i++)
        char_array[i] = s.at(i);

    for (int i = 0; i < n; i++)
        cout << char_array[i];

    return 0;
}
```

#include <string.h> !!!

# Arreglos Bidimensionales

- Escriba un programa en C++ que permita ubicar en un tablero de ajedrez de dimensiones 8x8 una reina por fila. Donde el usuario ingresa la posición de la reina para cada fila y el programa despliega las reinas en el tablero.

- Ejemplo: 0, 3, 4, 0, 0, 7, 6,7

```
Q.......
...Q....
....Q...
Q.......
Q.......
.......Q
......Q.
.......Q
```

# Arreglos Bidimensionales

- Considerando el programa anterior, ahora cree una función que valide que ninguna reina esta ubicada en una misma columna.

- Ejemplo:

```
...Q....
....Q...
Q.......
.Q......
..Q.....
.......Q
......Q.
.....Q..
```

válida

```
Q.......
...Q....
....Q...
Q.......
Q.......
.......Q
......Q.
.......Q
```

invalida

# Arreglos

- Emilio es ingeniero. Emilio realiza estudios en base a mediciones. Emilio sabe que, para calcular el promedio de una serie de mediciones no se debe considerar ni en mayor ni el menor de los valores. Emilio es un buen ingeniero. Para que Ud. sea como Emilio, se le pide que realice un programa que permita el ingreso de 100 mediciones y calcule el promedio como los calcula Emilio.

# Arreglos

- Considere una sequencia de adn representada en un arreglo unidimensional, donde cada casilla representa una de las bases nitrogenadas del ADN: timina (T), citocina(C), guanina (G), adenina (A). Un científico descubrió que la presencia de sub-secuencias TTCC genera capacidades excepcionales para programar. Se le solicita que cree un código en C++ que permita ingresar una secuencia y contabilizar el número de secuencias que poseen dicho patrón. Cada vez que se encuentre el patrón debe avisar al médico la existencia de este en dicha secuencia.

- Considere que cada secuencia tiene un largo de  300 bases.

# Arreglos

- Un entrenador de futbol desea obtener estadísticas de sus jugadores de manera rápida y para ello se le solicita que escriba un programa que haga uso de arreglos  para llevar registro de las capacidades de los futbolistas. Cada arreglo representara los siguientes atributos: Edad, Altura, Peso ,# lesiones, #expulsiones.

- El entrenador debe poder consultar el número del jugador de mayor edad, el más bajo de ellos, el de mayor peso, el con menos lesiones, y el con más expulsiones.

- Considere que los futbolistas se enumeran del 0 al 10. Donde el jugador 0 es el arquero.

# Arreglos Bidimensionales

- Genere un programa en C++ que permita la creación de una matriz  de tamaño N X N, con N ingresado por el usuario, en cuya diagonal superior este llena con 1's y en la diagonal inferior se llena con 0's.

| | | | |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 |

# Ejercicio

- Considere un arreglo unidimensional vacío. Llenar con número y luego encontrar repeticiones de los números contando el número de estas.

# Control

```
#include <iostream>

using namespace std;

int suma(int c){
        return *(&c) + 5;
}

void multiplica(int *d){
        *d = *d * 2;
}
```

```
int main(){
 int a = 5;
 int *b = new int;

 *b = a * 2;

 cout << suma(a) + *b << endl;
 multiplica(b);
 cout << *b << endl;

 cout << &a << endl;
 cout << b << endl;
}
```

- ¿Qué imprimen el primer par de cout?
- ¿Qué imprimen el segundo par de cout?
- Lo que imprime el segundo par de cout ¿es lo mismo?

# Control: Arreglos

- Un sistema de cultivo posee 3 sensores de temperatura que capturan datos cada 10 minutos. Los sensores evalúan cada 30 min si entran en alerta o no, para ello almacenan sus mediciones locales en un arreglo el cual consta de N registros por cada periodo 30 min. Una alerta se desencadena si el promedio de las temperaturas almacenadas  en los 30 min por el sensor es mayor a 25°.

- **Se solicita realizar un programa en C++ que controle las alertas para cada sensor. En caso de existir 2 o más alertas activas el programa debe arrojar un mensaje de peligro al usuario. Para efectos de este problema se asume que el usuario ingresa N registros de temperatura para cada uno de los 3 sensores. Utilice funciones y arreglos para la resolución del problema.**

# Solemne 2018

Garth Veyder, agente connotado del mal tiene varios prisioneros Yedai en celdas contiguas. Las celdas están organizadas en forma de un rectángulo de dimensiones N x M. Además, el valor de cada prisionero se mide por su conocimiento de la "Tensión" (es un primo de la Fuerza): a mayor conocimiento, mayor valor. Dicho valor se expresa como un valor real. Los prisioneros están ubicados arbitrariamente en las celdas y todas las celdas están ocupadas. De entre los Yedai capturados, algunos son yetas (tienen un valor de conocimiento inferior a 0). Ud., Maniquí Caminacielo, es un gran aliado de Veyder y un maestro interrogador. Veyder sabe que puede ser peligroso que él mismo trate con los Yedai y se encuentre con un yeta. Confiando en la experticia suya, le ha solicitado que interrogue a los Yedai empezando en la celda (0,0) y vaya "extrayendo" el conocimiento de los prisioneros. Sobre esto, Veyder le pide que mediante programas o funciones en C++:

a. Le entregue un informe de todas las ubicaciones de los Yedais yetas, de modo que pueda ir directamente a torturar a aquellos que no lo son. **(15 ptos)**
b. Le indique el monto total de conocimiento de la "Tensión" que hay entre todos los prisioneros. Para estos efectos, debe obviar a los Yedai yetas. **(15 ptos)**
c. Le entregue la ubicación del Yedai más experto en la "Tensión". **(15 ptos)**
d. Le entregue la ubicación del Yedai más yeta. **(15 ptos)**

Puede suponer que las asignaciones de celdas ya están hechas y no debe "llenarlas"

# Ejercicio

- Considere una matriz en que cada casilla representa el nivel del suelo en ese lugar. Se le solicita calcular cuántas unidades se requieren para nivelar toda la matriz con material, de manera de poder construir sobre un área nivelada.

# Ordenamiento

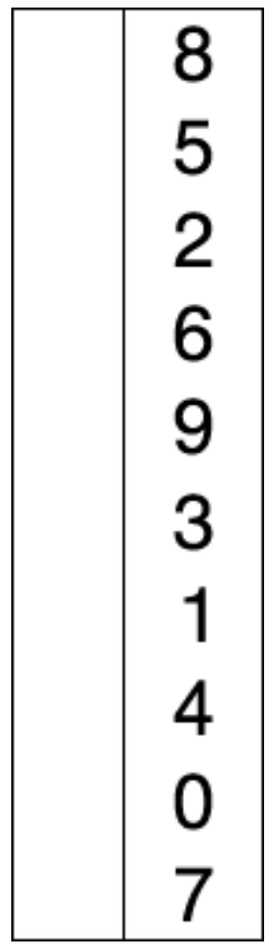
- **Burbuja:** ordenamiento de costo **O(n²)** que compara dos ítems adyacentes de un arreglo y las intercambia si ellos están en el orden equivocado.

- Idealmente, el proceso se repite hasta que no se requieren de más cambios

6 5 3 1 8 7 2 4

# Ordenamiento

- **Selection:**
  - ordenamiento de datos cuadrático **O(n²),** mismo orden de complejidad que bubble.
  - Algoritmo general consiste en seleccionar una casilla inicial y buscar el mínimo posible en todo el arreglo de datos. Cuando se encuentra se intercambia por el valor de la posición seleccionada.
  - Para un arreglo de tamaño **n** el costo siempre será **n²**



# Ordenamiento

- **Insertion:**
  - Ordenamiento de datos cuadrático **$O(n^2)$,**
  - A partir de un arreglo desordenado se genera un arreglo ordenado
  - Se selecciona un elemento y lo inserta en el arreglo en la posición que deja el arreglo parcialmente ordenado
  - Muy eficiente para arreglos pequeños y parcialmente ordenados

6 5 3 1 8 7 2 4

# Comparación

- *Selection* es menos costoso que el *bubble*, realiza menos swap o intercambios que su contraparte

- Una desventaja de *selection* es que no es capaz de detectar un arreglo ordenado, por lo tanto su costo es siempre el mismo

- Para el caso del *insertion*, es muy similar al *selection*, sin embargo solo compara con un subconjunto de los elementos del arreglo para insertar

# Ordenamiento

- **Ejercicio 7:** implementar el algoritmo *Bubble* (o burbuja) en C++ y ordenar un arreglo de 10 elementos.

- Implementar función sort (int*array, int n){}

# Ordenamiento

- **Ejercicio 7:** implementar el algoritmo *bubble* en C++ y ordenar un arreglo de 10 elementos.

- **¿Cómo podríamos hacerlo más eficiente?**

# Ordenamiento

- **Ejercicio 8:** implementar el algoritmo *Selection* en C++ y ordenar un arreglo de 10 elementos.

# Ordenamiento

- **Ejercicio 9:** implementar el algoritmo *Insertion* en C++ y ordenar un arreglo de 10 elementos.

# Ejercicio

**P3.-) (60 Puntos Total)** Después de una ardua investigación en genética, uno de los profesores del curso de Programación ha determinado que... los estudiantes que vienen a la UDP tienen una predisposición genética a ello y son muy selectos!!! Lo que observó el profesor en su investigación es que dados los cromosomas de la madre y del padre del estudiante es posible determinar la predisposición. En particular, se utiliza un alfabeto que resume el cromosoma a tamaño 10 y consta de todas las letras minúsculas del alfabeto español. La predisposición se da cuando se encuentra la secuencia "udp" ORDENADA Y USANDO ALGUNA COMBINACION DE LOS DOS CROMOSOMAS de los padres al menos una vez. Por ejemplo:

Caso 1, el estudiante no presenta predisposición:
Cromosoma de la madre: aty**d**bb**u**mlk
Cromosoma del padre: oolkinnahb

Caso 2, el estudiante presenta predisposición:
Cromosoma de la madre: cqb**u**lsbn**p**m
Cromosoma del padre: jxtzfh**d**ikh

Caso 3, el estudiante presenta predisposición:
Cromosoma de la madre: isygwolvfr
Cromosoma del padre: t**u**nz**d**rw**p**oj

**Escriba una función bool predisp(char madre[10], char padre[10]) que retorne si es que el estudiante tiene predisposición a ser un miembro de la 0s0m UDP. (30 ptos)**

# Ejercicio Clase

- **Ejercicio 10**: en la actualidad la mayoría de las plataformas computacionales constan de un sistema de cache. Un cache es un espacio de memoria rápido y de tamaño limitado donde almacenamos datos de uso frecuente. Utilizando arreglos genere un sistema de cache que permita mantener solo mantener las últimas  5 palabras más consultadas. Para reemplazar un valor se utiliza la política de la palabra con menos hits, donde un hit corresponde a el uso de la entrada de cache.

- Ejemplo:  cache de tamaño 3, si llega una nueva palabra como “Campeón” se reemplaza “Copa” y se ingresa “Campeón” 1. Recuerde que hay que mantener el cache ordenado.

“Chile” 5

“Confederaciones” 4

“Copa” 2

# Ejercicio 1

- Se solicita modificar el programa para que cuente los pares e impares mayores que 0.

```
#include <iostream>

using namespace std;

void contar(float arr[], int largo){
  int pares = 0 , impares = 0;

  if(arr % 2 == 0 ){
       pares++;
     return pares;

  }else{
       impares++;
     return impares;
  }

}
```

```
int main(){
int  impar=0, par=0;

int arr[]={1,2,3,0,5,0,-1,4};

contar(arr, 8);

cout << impar << " "<< par << endl;

return 0;
}
```

# Resolución 1

```
#include <iostream>

using namespace std;

void contar(int* arr, int largo, int *impar, int * par){

 for (int i=0; i < largo; i++){

        if(arr[i] >0 && arr[i]% 2 == 0 ){
                *par= *par+1;
        }else{
                if(arr[i] >0)
                        *impar= *impar +1;
        }
 }
}

int main(){
int impar=0, par=0;

int arr[]={1,2,3,0,5,0,-1,4};

contar(&arr[0], 8, &impar, &par);

cout << par << " "<< impar<<endl;

return 0;
}
```

# Resolución 2

```
#include <iostream>

using namespace std;

int * contar(int* arr, int largo){

 int *tmp= new int[2];
 tmp[0] = 0;
 tmp[1] = 0;

 for (int i=0; i < largo; i++){
        if(arr[i] > 0 && arr[i] % 2 == 0 ){
                tmp[0]++; // par

        }else{
                if(arr[i] > 0){
                        tmp[1]++; // impar
                }
        }
 }

 return tmp;
}
```

```
int main(){
int *ptr;

int arr[]={1,2,3,0,5,0,-1,4};

ptr = contar(&arr[0], 8);

cout << ptr[0] << " "<< ptr[1]<<endl;

return 0;
}
```

# Otros ejemplos…

```
#include <iostream>

using namespace std;

int * contar(int **arr, int largo, int ancho){

 int *tmp=new int[2];
 tmp[0]=0;
 tmp[1]=0;

 for (int i=0; i < largo; i++){
        for(int j =0 ; j< ancho;j++){
         if(arr[i][j] > 0 && arr[i][j] % 2 == 0 ){
                tmp[0]++; // par

        }else{
                if(arr[i][i] > 0){
                        tmp[1]++; // impar
                }
        }
    }

 }

return tmp;
}
```

```
int main(){

int *ptr;

//creando array de punteros
int **array = new int*[2];
for( int i = 0; i < 2; ++i )
    array[ i ]  = new int[ 2 ];

//llenado
for(int i = 0; i <2 ; i++){
        for(int j = 0; j <2 ; j++){
                if(i == j)
                        array[i][j] = 1;
                else
                        array[i][j] = 2;
        }
}

ptr = contar(array, 2,2);
cout << ptr[0] << " "<< ptr[1]<<endl;

return 0;
}
```

# Ejercicio 2

- Emilio es ingeniero. Emilio realiza estudios en base a mediciones. Emilio sabe que, para calcular el promedio de una serie de mediciones no se debe considerar ni en mayor ni el menor de los valores. Emilio es un buen ingeniero. Para que Ud. sea como Emilio, se le pide que programe una función que retorna el promedio como los calcula Emilio.

- **float promedio(float arr[], int largo){ }**

*Te pillamos compadre!*

# Resolución 2

```
#include<iostream>

using namespace std;

void sort(float array[], int size){
int tmp;
bool swapped;

do{
    swapped = false;
    for (int j=0; j < size-1; j++){
         if(array[j] > array[j+1]  ){
                  tmp=array[j];
                  array[j] = array[j+1];
                  array[j+1] = tmp;
                  swapped = true;
         }
     }
  }while(swapped);
}
```

```
float promedio(float arr[], int largo){

  float suma=0;
  sort(arr,largo);

  for(int i = 1; i< largo -1 ; i++){
        suma= arr[i] + suma;
  }

    return suma/(float)(largo-2);

}

int main(){

  float arr[] = {7.0,1.5,4.0,2.5};
  float prom;
  prom = promedio(arr, 4);
  cout << prom<< endl;
}
```

# Archivos de texto

- #include <fstream>
- **Abrir Archivo:**
  - ifstream myfile;
  - myfile.open(“la_ruta_del_archivo.algo”)
  - Ojo solo para lectura!!! **READ ONLY**
- **Cerrar Archivo:**
  - myfile.close();

# Archivos de texto

- #include <fstream>
- **Abrir Archivo:**/
  - ofstream myfile;
  - myfile.open(“la_ruta_del_archivo.algo”)
  - Ojo, ahora modo escritura!!  borra todo el contenido del archivo si este existe!
- **Cerrar Archivo:**
  - myfile.close();

# Archivos de texto

- #include <fstream>
- **Abrir Archivo:**
  - fstream myfile;
  - myfile.open(“la_ruta_del_arcchivo.algo”)
  - Ojo, este acepta ambos métodos, lectura/escritura.
- **Cerrar Archivo:**
  - myfile.close();

# Archivos

- **Resumen modos de apertura:**

| | |
|---|---|
| `ios::in` | Open for input operations. |
| `ios::out` | Open for output operations. |
| `ios::binary` | Open in binary mode. |
| `ios::ate` | Set the initial position at the end of the file.<br>If this flag is not set, the initial position is the beginning of the file. |
| `ios::app` | All output operations are performed at the end of the file, appending the content to the current content of the file. |

# Archivos

```
#include<iostream>
#include<fstream>

using namespace std;

int main(){
fstream myfile;
myfile.open("salida.o",ios::app); //Opciones in,out,app

if(myfile.is_open()){
        myfile << "Esto es un ejemplo de como escribir en un archivo"<<endl;
        myfile << "Si quiero una segunda linea, hago esto"<<endl;
        myfile.close();

}else{
        cout<< "Error al abrir el archivo...!"<<endl;
}


return 0;
}
```

# Búsqueda

- **tellg() y tellp():** permiten recuperar la posición del cursor (*streampos*) en el archivo.

- **seekg() y seekp():** permiten cambiar la posición del cursor en el archivo para realizar operaciones get y put.

```
seekg ( position );      seekg ( offset, direction );
seekp ( position );      seekp ( offset, direction );
```

| | |
|---|---|
| `ios::beg` | offset counted from the beginning of the stream |
| `ios::cur` | offset counted from the current position |
| `ios::end` | offset counted from the end of the stream |