

Algoritmos y Estructuras de Datos

Oscar Meza

omezahou@ucab.edu.ve

Ejercicio 1:

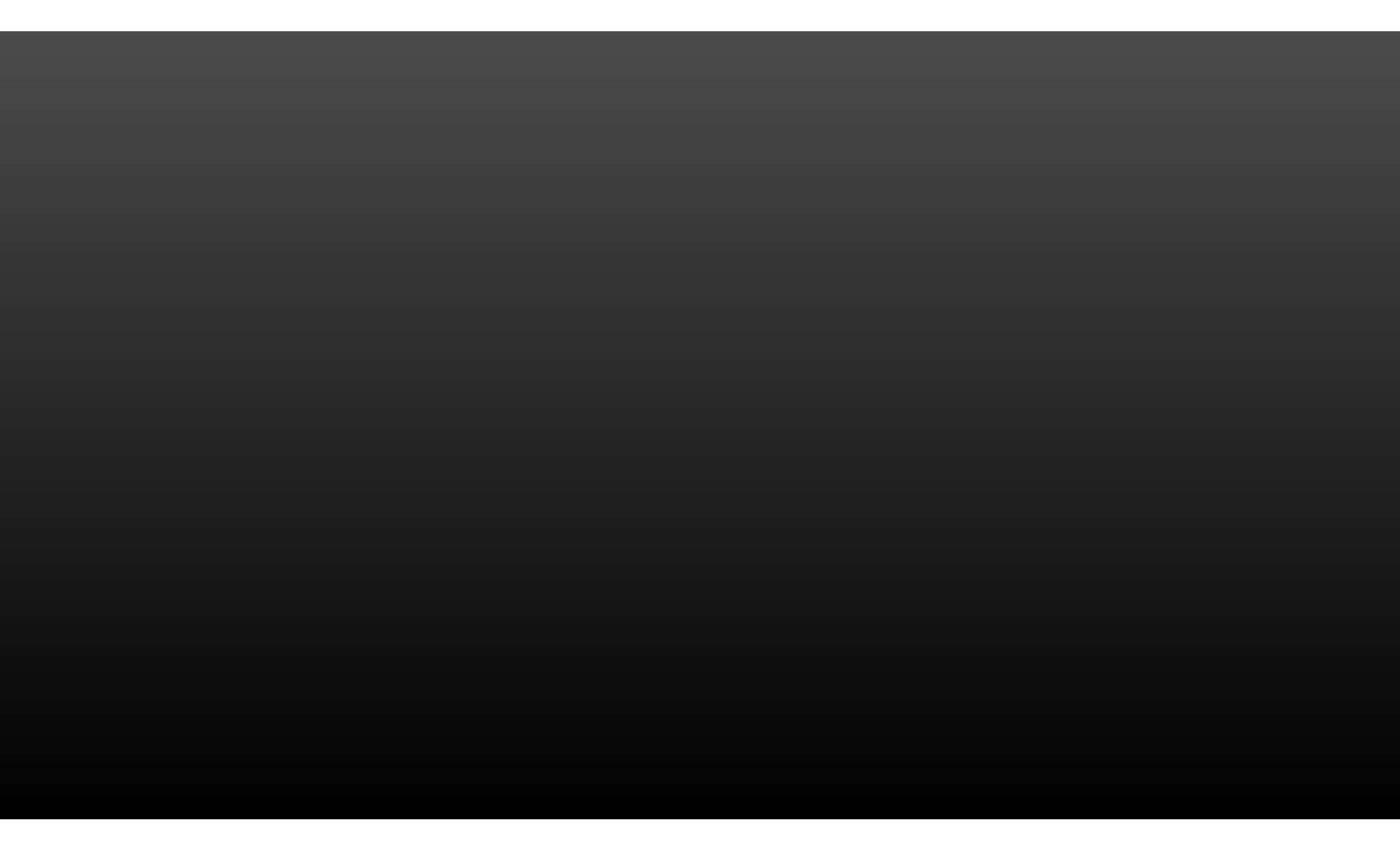
Vamos implementar en C++ con header (.h) y .cpp una el tipo Conjunto

Para esto:

- Crear una carpeta vacía llamada Conjunto (**NO PUEDE HABER OTRO .CPP DE OTROS PROGRAMAS, SOLO LOS DE CONJUNTO**).
- Bajarse a esta carpeta, los archivos Conjunto.h, Conjunto.cpp, prueba_conjunto.cpp de Modulo 7 (laboratorio 3) Ejecutar VSCODE y abrir esa carpeta (open folder)
- Ejecutar prueba_conjunto.cpp (DARA ERROR) pero crea carpeta vscode y dentro el archivo tasks.json
- **cambiar el tasks.json : "\${file}", por "\${workspaceFolder}*.cpp", si quiere ejecutar el programa con VScode. Si no utilice el comando en terminal:**
 - **g++ -o prueba_conjunto prueba_conjunto.cpp Conjunto.cpp**

Agregue a la implementación dada del TDA conjunto de enteros (modificando los archivos .h y .cpp de Conjunto) :

- la operaciones de intersección de dos conjuntos y devolver un conjunto nuevo con la intersección,
- la operación de diferencia de dos conjuntos y devolver un conjunto nuevo con la diferencia.
- Una función que devuelva los elementos de un conjunto en un vector, para poder tener acceso a los elementos del conjunto
- Y luego un programa de prueba para probar el tipo Conjunto.



```
Conjunto intersectar(Conjunto A, Conjunto B){
    Conjunto result;
    for (auto i : A)
        if (pertenece(B , i)) result.push_back(i);
    return result;
}
```

```
Conjunto diferencia(Conjunto A, Conjunto B){
    Conjunto result;
    for (auto i : A)
        if (! pertenece(B , i)) result.push_back(i);
    return result;
}
```

```
vector<int> elementos(Conjunto A){
    return A;
}
```

Ejercicio 2:

Numeros de Fibonacci: $Fib(0)=0$ $Fib(1)=1$, $Fib(n) = Fib(n-1) + Fib(n-2)$ $n \geq 2$

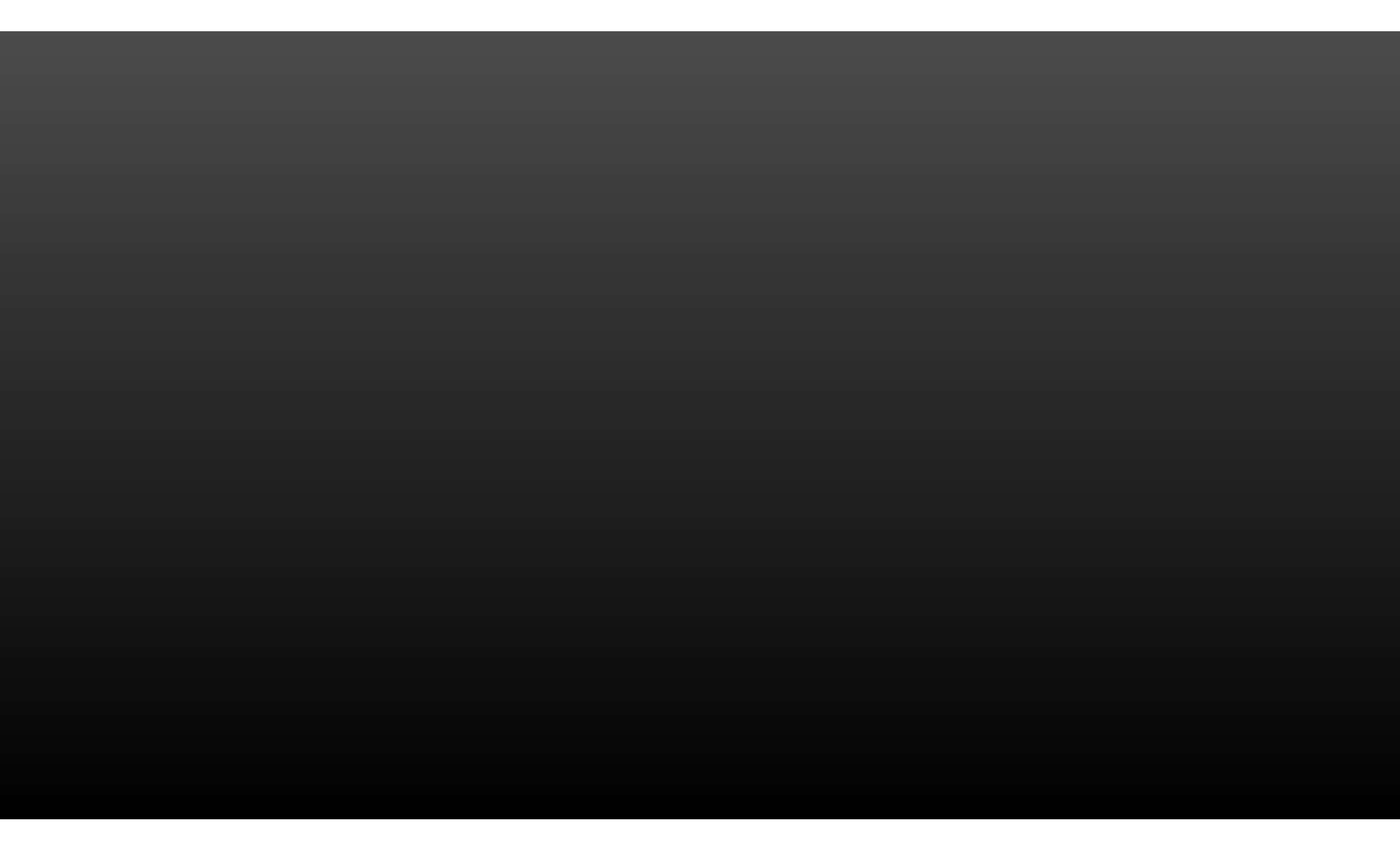
- Crear una carpeta Fibonacci
- Implementar las versiones iterativa y recursiva para calcular $Fib(n)$
Prototipo: `long Fib_iter(int n)` y `long Fib(int n)`
- comparar los tiempos de cálculo recursivo e iterativo de los números de fibonacci para $n = 40$ a 50
- Utilice esta biblioteca que es mas precisa para medir el tiempo.....
Ver siguiente lámina.....

```
#include <chrono>
#include <iomanip> // para formato de salida (right, setw)

auto comienzo= std::chrono::system_clock::now();
long fib_itr = fibonacci_itr(n);
auto fin = std::chrono::system_clock::now();
std::chrono::duration<float,std::micro> duracion
    = fin - comienzo;

// Imprimir duracion

std::cout << "Iterativo: Para n = "
    << std::right<<std::setw(3)<<n << " Fib(n) = "<< fib_itr
    <<" Tiempo de ejecucion = " << duracion.count()
    << " microsegundos" << '\n';
```



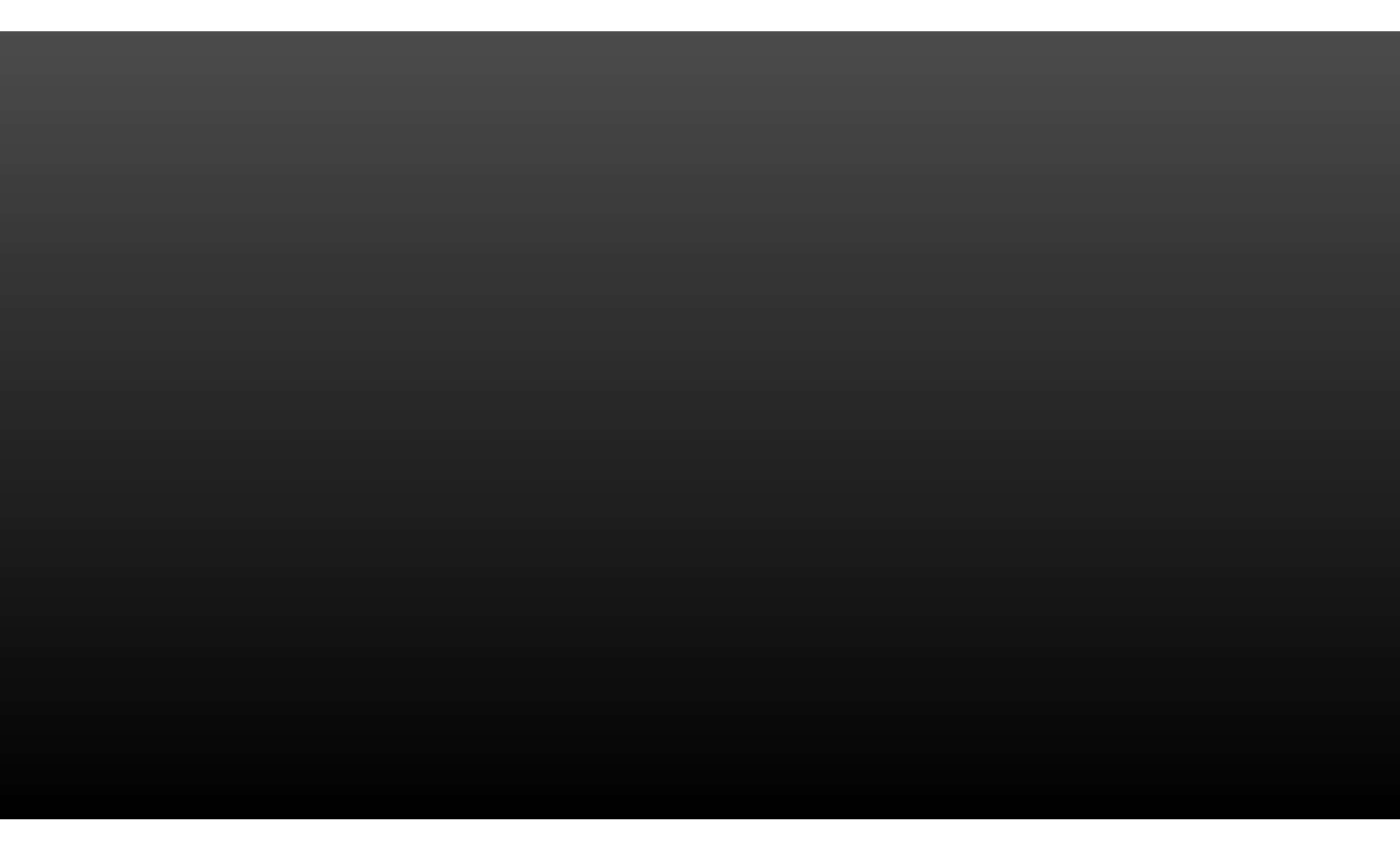

```
long fibonacci_itr( int n )
{
    // n >= 0
    int fn_1 = 1, fn_2 = 0,
        fib = 0 ;
    if (n==0 || n==1) return n;
    for (int i = 2; i<=n; i++) {
        fib = fn_1 + fn_2;
        fn_2 = fn_1;
        fn_1 = fib;
    }
    return fib;
}
```

```
long fib_rec( int n )
{
    // n >= 0
    if( n <= 1 ) return n;
    else
        return fib_rec(n - 1)
            + fib_rec(n - 2);
}
```

Ejercicio 3:

Hacer un algoritmo recursivo tal que dado un entero x , determine si existe un entero i tal que $A[i] = x$ en un arreglo de enteros de largo N , ordenado en forma creciente (por búsqueda binaria)

Ejemplo: buscar $x = 11$ en 5, 8, 8, 9, 10, 11, 16, 16



```
#include <iostream>
#include<vector>

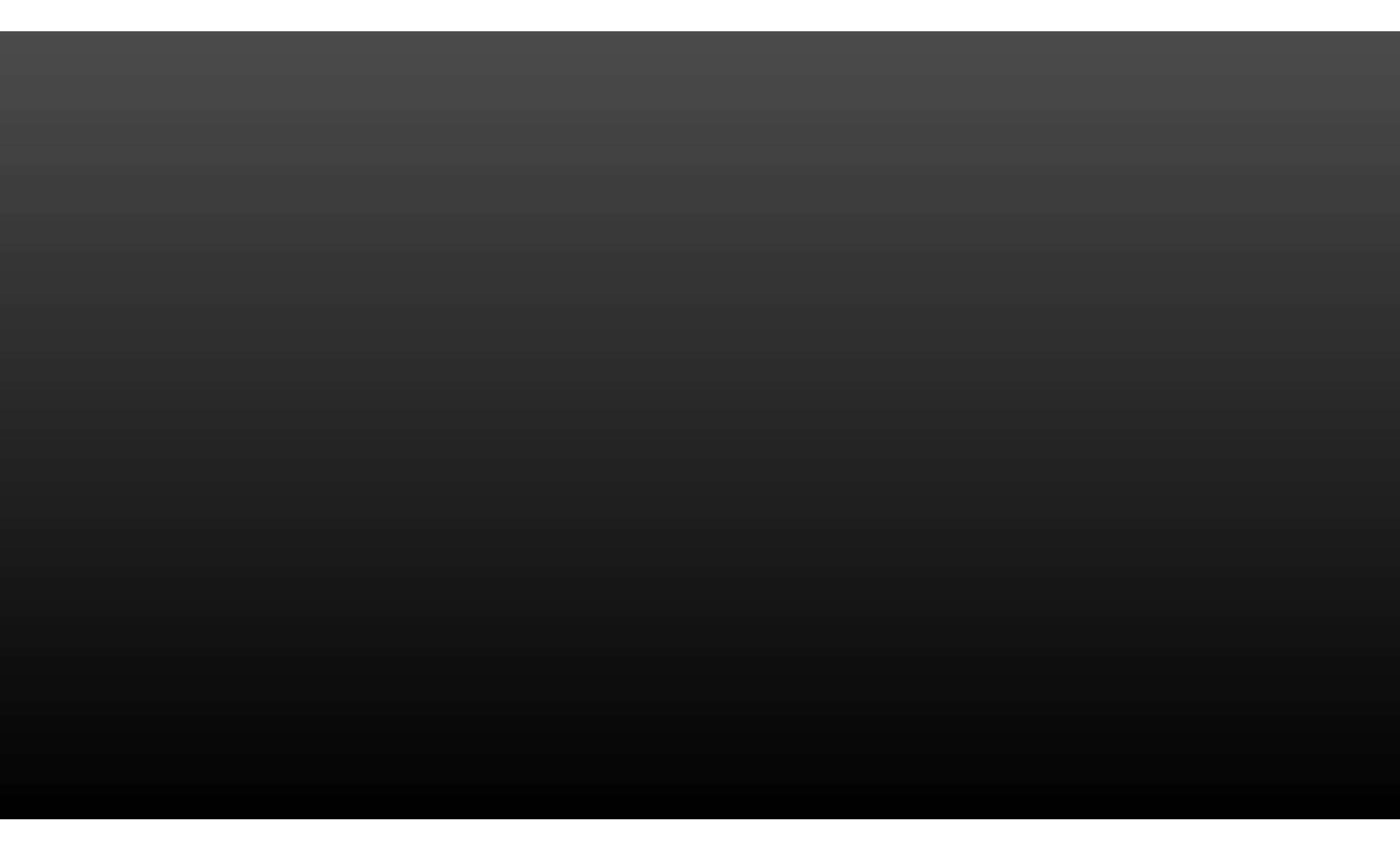
bool Buscar (vector<int> a,int clave){ //driver
    return busquedaBinaria(a,4,0,a.size());
}

int main(){
    vector<int> v {1,3 ,5 ,7};
    std:: cout << "es: " << Buscar(v,4)
        << std :: endl;
}
```

```
bool busquedaBinaria(vector<int> a,int clave, int inf, int sup){
    int central;
    if(inf > sup)
        return false;
    else{
        central = (inf + sup)/2;
        if (a[central] == clave)
            return true;
        else if (a[central] < clave)
            return busquedaBinaria(a,clave,central + 1, sup);
        else
            return busquedaBinaria(a,clave,inf, central -1);
    }
}
```

Ejercicio 4:

Convertir Búsqueda Binaria recursivo en iterativo



```
bool busquedaBinaria_itr(vector<int> a,int clave, int inf, int sup){
    int central;
    while(inf<=sup){
        central = (inf + sup)/2;
        if (a[central] == clave)
            return true;
        else if (a[central] < clave)
            inf= central + 1 ;
        else
            sup = central - 1;
    }
    return false;
}
```


Ejercicio 5: Usando recursividad

- Imprimir todos los vectores 0-1 de largo n
- Imprimir todos los subconjuntos de un conjunto usando lo anterior y colocarlo como funcion de Conjunto.cpp en carpeta Conjunto
- Imprimir todos los subconjuntos de tamaño m de un n-conjunto usando lo anterior (evitar generar todos los subconjuntos)



```
void imprimir_subconjuntos(Conjunto A){
    vector<int> cero_uno;
    cero_uno.clear();
    imprimir_sub_rec(0,A,cero_uno);
}
```

```
void imprimir_sub_rec(int i, Conjunto A,
                    vector<int> cero_uno){
    // se generan vectores cero uno que
    // representan cada subconjunto
    if (i==A.size()) {
        cout<<"{ ";
        for (int i = 0; i<A.size(); i++){
            if (cero_uno[i]) cout<<A[i]<<" ";
        }
        cout<<"}";
        cout<<endl;
        return;
    }
    cero_uno.push_back(1); // comienzan con uno
    imprimir_sub_rec(i+1,A,cero_uno);
    cero_uno.pop_back(); // quitar el uno
    cero_uno.push_back(0); // comienzan con cero
    imprimir_sub_rec(i+1,A,cero_uno);
    cero_uno.pop_back(); // quitar el cero
    // no es necesario quitar el cero
    // por ser llamada por valor
}
```

Ejercicio 6:

Hacer un programa que lea un conjunto de estudiantes: nombre, apellido, cedula, edad, promedio (índice académico) y cree un archivo con esta información.

Luego lea el archivo y lo coloque en un vector y busque el estudiante con el mayor promedio.

Archivos de texto de lectura y escritura

Como crear un Archivo de texto de salida (crear un archivo de texto):

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

int main() {
    int entero;
    std::string filename;
    std::cout << " Nombre del archivo:";
    std::cin >> filename;
    // crear archivo
    std::ofstream out (filename); //crea out y lo
    // asocia al archivo (lo abre)

    if (out.is_open()) {    // si se abrio bien el archivo
        // colocar varios enteros y terminar con un caracter
        while (std::cin>>entero){
            // termina si coloco un carácter,
            // lo hace por un error al leer un caracter.
            //Habría que "resetear "cin" para utilizarlo después
            out << entero << " ";
        }
    } else    std::cout << "No se pudo escribir\n";
    out.close();
    return 0;
}
```

Leer un Archivo de texto de entrada (leemos un archivo):

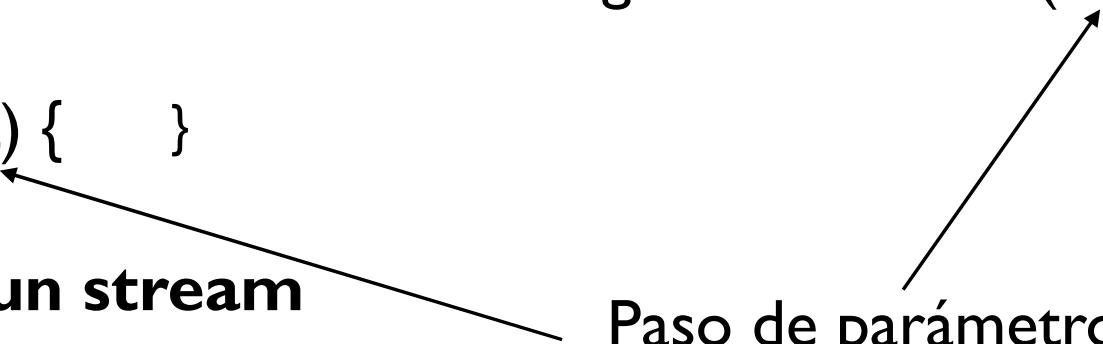
```
std::cout << " Coloque el nombre del archivo: ";
std::cin >> filename;
std::ifstream in (filename);
int value;
if (in.is_open()) { // verificar archivo abierto
    while (in >> value) // leer hasta fin de archivo
        std::cout << value << "\n";
}
else std::cout << "No se puede abrir el archivo\n";
in.close();
```

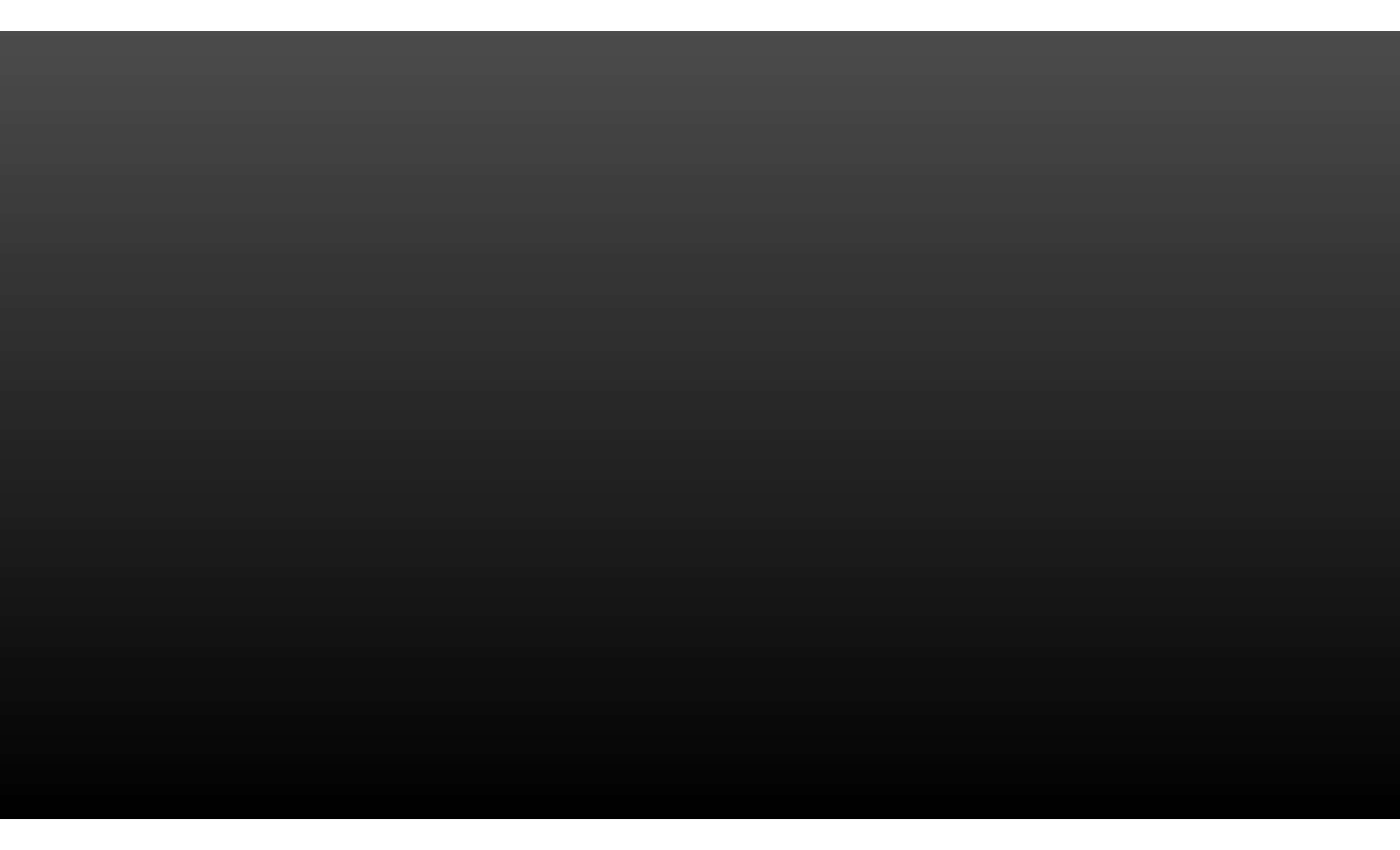
Dividir en palabras, una cadena de caracteres leída con espacios:

```
#include <iostream>
#include <string>
#include <vector>
#include <sstream>
using std::vector;
using std::string;
vector<string> Extraer(const string Text) {    }
    vector<string> Words;
    std::stringstream ss(Text); // ss es un stream
    string Buf;
    while (ss >> Buf) // mientras existan palabras
        Words.push_back(Buf);
    return Words;
}
```

```
int main() {
    string linea ;
    std::cout << "Coloque una linea de texto: ";
    getline(std::cin, linea);
    vector<string> xx = Extraer(linea);
}
```

Paso de parámetro por VALOR





```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>

struct estudiante{
    std::string nombre;
    std::string apellido;
    std::string cedula;
    int edad;
    double promedio;
};

estudiante mayor_promedio(std::vector<estudiante> v){
    estudiante est_mayor_promedio ;
    est_mayor_promedio.promedio=-1;
    for (auto est: v)
        if (est.promedio > est_mayor_promedio.promedio)
            est_mayor_promedio = est;
    return est_mayor_promedio;
}
```

```
int main() {
    std::string line;
    std::string filename;
    std::cout
    << " Coloque el nombre del archivo: ";
    std::cin >> filename;
    // cin queda con una linea vacia que hay
que leer
    getline(std::cin,line);

    // crear archivo y guardar linea a linea
leida
    // en el archivo
    std::ofstream out(filename);
```

```
// ingrese los numeros a ser guardados en el archivo

if (out.is_open()) { // si se abrio bien el archivo
    // colocar nombre apellido cedula edad promedio y enter
    std::cout << " Coloque las lineas una a una y al final una linea vacia: \n";
    getline(std::cin,line);
    // note que al final del archivo habra una linea vacia
    // debido al "\n" de la ultima linea a ingresar
    while ( line!="" ){
        out << line << "\n";    // se creara una linea vacia al final
        getline (std::cin,line);
    }
}
else
    std::cout << "No se pudo escribir en el archivo\n";

out.close();
```

```
// LEER ARCHIVO Y COLOCARLO EN ARREGLO DE ESTUDIANTES
std::cout << " Coloque el nombre del archivo: ";
std::string text;
std::cin >> filename;
std::ifstream in(filename);

std::vector<estudiante> v;
estudiante var;
while(!in.eof())
{
    in>>var.nombre;
    in>>var.apellido;
    in>>var.cedula;
    in>>var.edad;
    in>>var.promedio;
    v.push_back(var);
}
v.pop_back(); //OJO: porque el ultimo elemento de in es una linea vacia
               // entra al while no lee nada y guarda en el vector
               // los valores anteriores de var (el ultimo estudiante)
estudiante est = mayor_promedio(v);
}
```