

UNIVERSIDAD DEL NACIONAL DEL ALTIPLANO

**FACULTAD DE MECANICA ELECTRICA, ELETRONICA Y SISTEMAS
ESCUELA PROFESIONAL DE INGENIERIA DE SISTEMAS**



PROYECTO FINAL METODOS DE ORDENAMIENTO EN C++

CURSO:

PENSAMIENTO COMPUTACIONAL

DOCENTE:

ING. MIGUEL ROMILIO ACEITUNO ROJO

PRESENTADO POR:

- MIRANDA JOSEC YEFFERSON
- RAMOS JAHUIRA GABRIEL ANDERSON
- RAMOS CASTILLO JEFRY EDINSON

PUNO-PERU

2024

[illegible]

```

1  #include <iostream>
2  using namespace std;
3
4  void selectionSort(int arr[], int size, bool ascending, int &cycles)
5  {
6      for (int i = 0; i < size - 1; ++i)
7      {
8          int minIndex = i;
9          for (int j = i + 1; j < size; ++j)
10             if ((ascending && arr[j] < arr[minIndex]) || (!ascending && arr[j] > arr[minIndex]))
11                 minIndex = j;
12             cycles++;
13             swap(arr[minIndex], arr[i]);
14         }
15     }
16 }
17
18 // Implementación de la función de ordenamiento countingSort
19 void countingSort(int arr[], int size, bool ascending, int &cycles)
20 {
21     int maxElement = *max_element(arr, arr + size);
22     int minElement = *min_element(arr, arr + size);
23     int range = maxElement - minElement + 1;
24     int count[range] = {0};
25
26     for (int i = 0; i < size; ++i)
27     {
28         count[arr[i] - minElement]++;
29         cycles++;
30     }
31 }

```

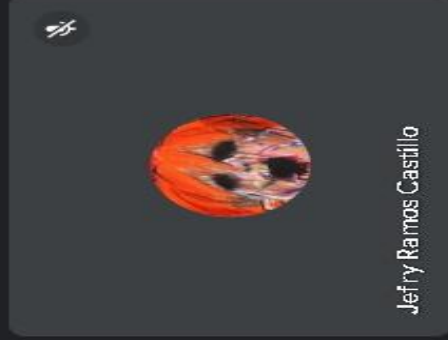
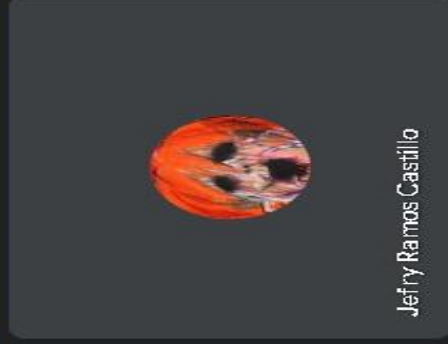
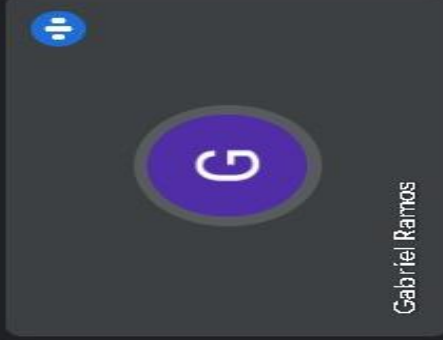
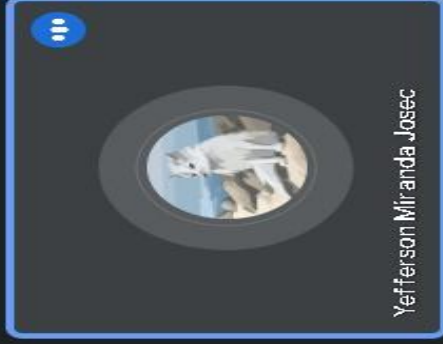
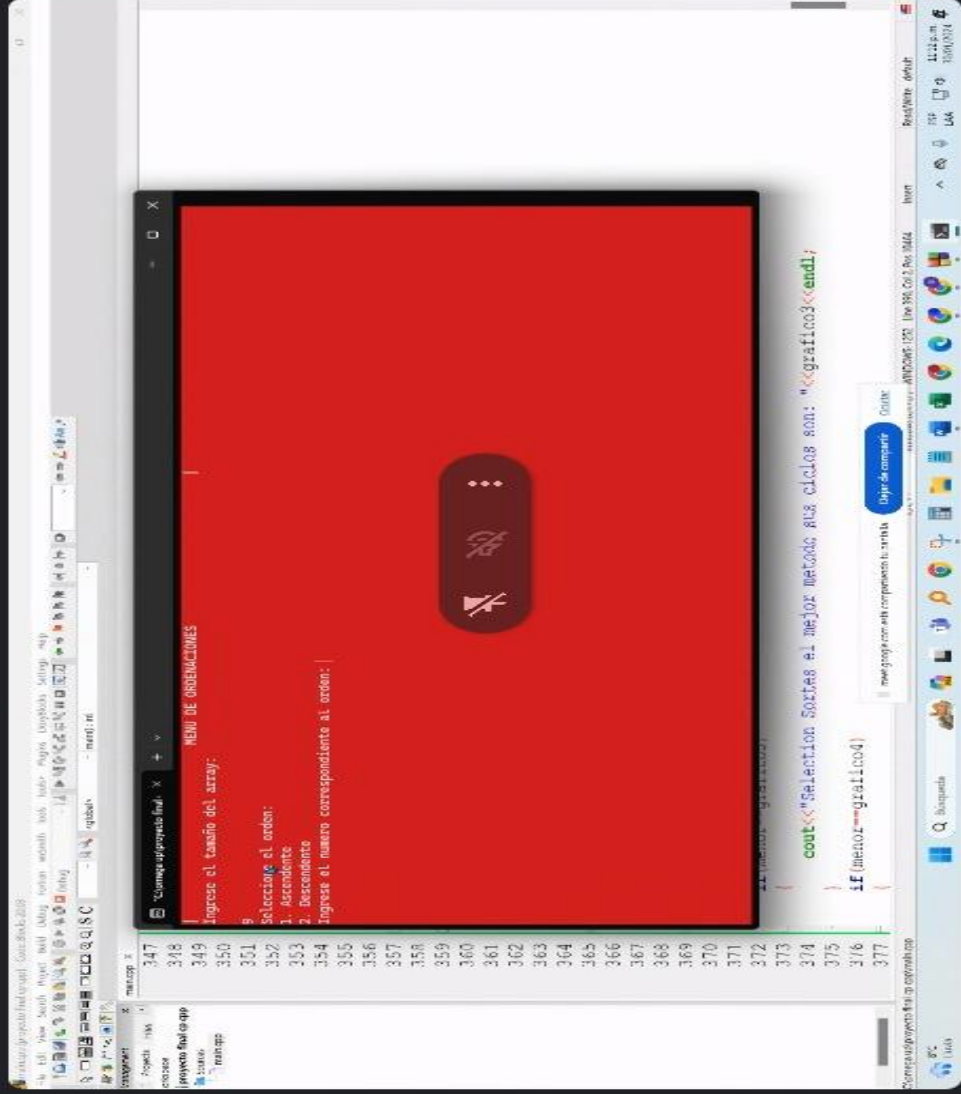
The diagram illustrates a network structure with four nodes arranged in a 2x2 grid. Each node is represented by a circular icon and a corresponding name. The top-left node features a purple circle with a white letter 'G' and is labeled 'Gabriel Ramos'. The top-right node features a red skull icon and is labeled 'Jeffry Ramos Castillo'. The bottom-left node features a white dog icon and is labeled 'Jefferson Miranda Josec'. The bottom-right node features a red skull icon and is labeled 'Jeffry Ramos Castillo'. A blue border highlights the bottom-left node, and a blue circle with three dots is positioned above it, indicating a menu or options. The background is dark gray.

[illegible]

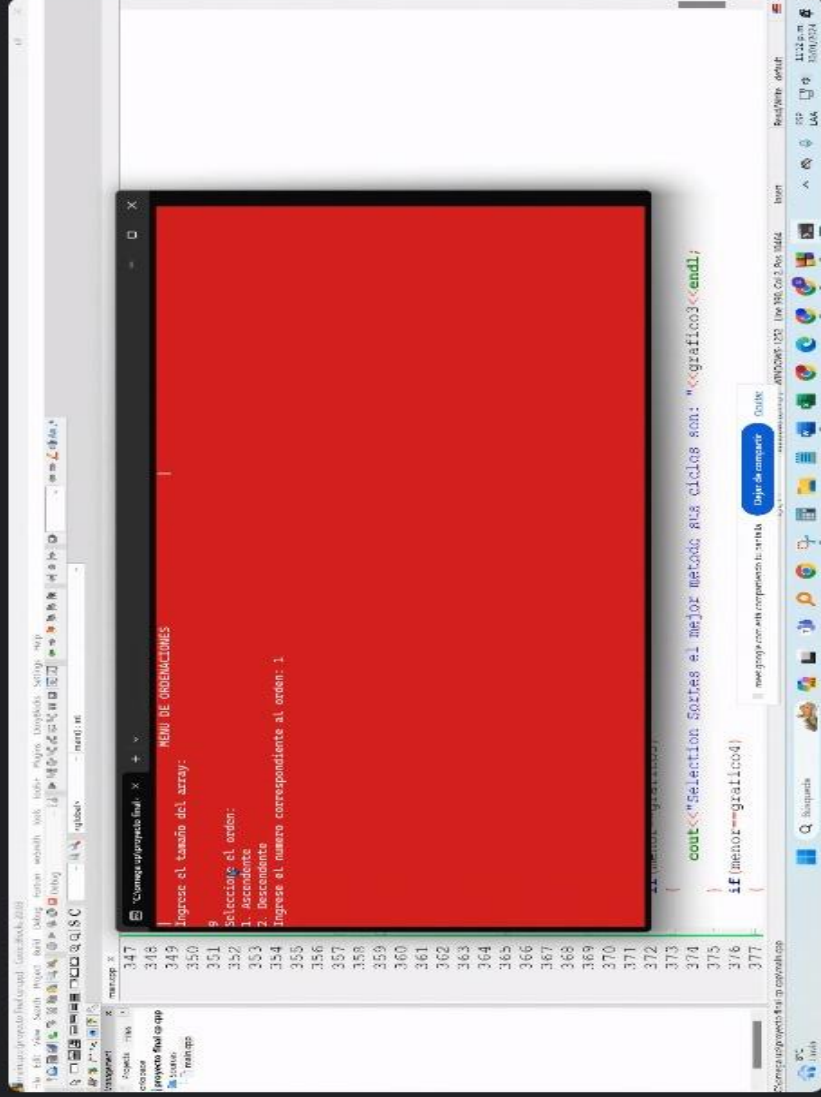
The image is a screenshot of a mobile screen, likely a tablet or smartphone, displaying a presentation and a code editor. The top status bar shows the time as 23:09, the date as 30/01/2024, and the battery level at 100%. The presentation slide, titled 'Gabriel Ramos (Presenter)', features a dark background with a grid of four cards. The top-left card has a purple circle with a white 'G' and the name 'Gabriel Ramos'. The top-right card has a circular image of a person's face and the name 'Jeffry Ramos Castillo'. The bottom-left card has a circular image of a person's face and the name 'Jefferson Miranda Josec'. The bottom-right card has a circular image of a person's face and the name 'Jeffry Ramos Castillo'. Below the presentation, a code editor window is open, showing C++ code for a merge sort algorithm. The code includes comments in Spanish and uses standard C++ syntax for arrays, recursion, and sorting. The code is as follows:


```
227 merge(arr, low, middle, high, ascending, cycles);  
228 }  
229 }  
230 // función principal  
231 int main()  
232 {  
233     system("Color 4p");  
234     //color de la consola  
235     system("Color 20");  
236     // tamaño del array  
237     int size;  
238     cout << "Intrúste el tamaño del array: ";  
239     cin >> size;  
240     if (size <= 0)  
241     {  
242         cout << "tamaño no válido. Saliendo del programa.\n";  
243         return 1;  
244     }  
245     // Crear el array  
246     int arr[size];  
247     // llenar el array con números aleatorios  
248     fill(arr, arr + size, size);  
249     // Contador de ciclos  
250     int cycles = 0;  
251     mergeSort(arr, 0, size - 1, true, cycles);  
252     cout << "El array está ordenado.\n";  
253     return 0;  
254 }
```


G Gabriel Ramos (Presentar)




Gabriel Ramos (Presentar)






Yefferson Miranda Josec




Fijar

Gabriel Ramos



Jeffry Ramos Castillo



Jeffry Ramos Castillo



Todos los favoritos

Gabriel Ramos (Presentar)

```
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377

// Selecciona el orden:
// 1. Ascendente
// 2. Descendente
// Ingrese el numero correspondiente al orden: 1
// Array original: 8 43 76 20 91 64 92 4 98
// Resultados de los metodos de ordenamiento:
// Metodo Array ordenado Ciclos
// Burbuja Sort: 8 4 20 43 64 76 90 91 93 36
// Insertion Sort: 8 4 20 43 64 76 90 91 93 12
// Selection Sort: 8 4 20 43 64 76 90 91 93 36
// Counting Sort: 8 4 20 43 64 76 90 91 93 18
// Quick Sort: 8 4 20 43 64 76 90 91 93 18
// Merge Sort: 8 4 20 43 64 76 90 91 93 29
// GRAFICO
// Burbuja Sort: #####
// Insertion Sort: #####
// Selection Sort: #####
// Counting Sort: #####
// Quick Sort: #####
// Merge Sort: #####
// Insertion Sort es el mejor metodo sus ciclos son: 12
// Process returned 0 (0x0): execution time : 75.387 s
// Press any key to continue.
cout<<"seleccion Sortes el mejor metodo sus ciclos son: "<<grafico3<<endl;
if (menor==grafico4)
```

Gabriel Ramos

Jefferson Miranda Josec

Jeffrey Ramos Castillo

Jeffrey Ramos Castillo



5



23:12 | ici-ymph-emm

Yefferson Miranda Josec



Gabriel Ramos



Jerry Ramos Castilla




Jerry Ramos Castilla

Gabriel Ramos (Presenter)

The image shows a Windows 10 desktop with a Visual Studio Code editor open. The editor displays a C# program that implements five sorting algorithms: Bubble Sort, Selection Sort, Counting Sort, Quick Sort, and Merge Sort. The program uses a console application structure with a 'Program' class and a 'Main' method. The output of the program is shown in the console window, indicating that the algorithms were executed on an array of numbers, and the final sorted array is 97 15 13 24 48 62 62 21 17. The background of the desktop is a red wall with a clock and a calendar. The taskbar at the bottom shows the Start button, a search bar, and several pinned applications including File Explorer, Visual Studio Code, and a web browser. The system tray in the bottom right corner shows the date and time as 11:25 AM on 10/24/2024.

The image displays two cards from a game, arranged horizontally. The left card features a circular image of a white dog, possibly a Weimaraner, standing on a sandy beach. Below the image, the name 'Yefferson Miranda Josec' is written in white text. The right card features a circular image of a jack-o'-lantern with a carved face. Below the image, the name 'Jeffry Ramos Castilla' is written in white text. Both cards have a blue circle with three white dots in the top right corner, indicating a menu or options button. The cards are set against a dark gray background.

G Mais opções



Jeffrey Ramos Castillo


```

#include <iostream>
#include <iomanip>
#include <cmath>
#include <ctime>
#include <cstdlib>
#include <algorithm>
#include <stdlib.h>
#include <locale.h>

#include <stdio.h>

using namespace std;
//grafiquito
void drawBar(int value) {
    // Utilizamos el carácter '#' para representar una barra en el
    gráfico
    for (int i = 0; i < value; ++i) {
        cout << '#';
    }
    cout << endl;
}

// Prototipos de funciones de ordenamiento
void bubbleSort(int arr[], int size, bool ascending, int &cycles);
void insertionSort(int arr[], int size, bool ascending, int &cycles);
void selectionSort(int arr[], int size, bool ascending, int &cycles);
void countingSort(int arr[], int size, bool ascending, int &cycles);
void quickSort(int arr[], int low, int high, bool ascending, int
&cycles);
void mergeSort(int arr[], int low, int high, bool ascending, int
&cycles);

// Función para imprimir el array
void printArray(int arr[], int size)
{
    for (int i = 0; i < size; ++i)
    {
        cout << arr[i] << " ";
    }
}

// Función para llenar el array con números aleatorios
void fillArrayRandom(int arr[], int size)
{
    srand(time(NULL));
    for (int i = 0; i < size; ++i)
    {
        arr[i] = rand() % 100;
    }
}

// Implementación de la función de ordenamiento bubbleSort
void bubbleSort(int arr[], int size, bool ascending, int &cycles)
{

```

```

    for (int i = 0; i < size - 1; ++i)
    {
        for (int j = 0; j < size - i - 1; ++j)
        {
            if ((ascending && arr[j] > arr[j + 1]) || (!ascending &&
arr[j] < arr[j + 1]))
            {
                swap(arr[j], arr[j + 1]);
            }
            cycles++;
        }
    }
}

// Implementación de la función de ordenamiento insertionSort
void insertionSort(int arr[], int size, bool ascending, int &cycles)
{
    for (int i = 1; i < size; ++i)
    {
        int key = arr[i];
        int j = i - 1;

        while ((j >= 0) && ((ascending && arr[j] > key) || (!ascending
&& arr[j] < key)))
        {
            arr[j + 1] = arr[j];
            --j;
            cycles++;
        }
        arr[j + 1] = key;
    }
}

// Implementación de la función de ordenamiento selectionSort
void selectionSort(int arr[], int size, bool ascending, int &cycles)
{
    for (int i = 0; i < size - 1; ++i)
    {
        int minIndex = i;
        for (int j = i + 1; j < size; ++j)
        {
            if ((ascending && arr[j] < arr[minIndex]) || (!ascending
&& arr[j] > arr[minIndex]))
            {
                minIndex = j;
            }
            cycles++;
        }
        swap(arr[minIndex], arr[i]);
    }
}

// Implementación de la función de ordenamiento countingSort
void countingSort(int arr[], int size, bool ascending, int &cycles)
{
    int maxElement = *max_element(arr, arr + size);
    int minElement = *min_element(arr, arr + size);

```

```

    int range = maxElement - minElement + 1;
    int count[range] = {0};

    for (int i = 0; i < size; ++i)
    {
        count[arr[i] - minElement]++;
        cycles++;
    }

    int outputIndex = 0;
    for (int i = 0; i < range; ++i)
    {
        while (count[i] > 0)
        {
            arr[outputIndex++] = i + minElement;
            count[i]--;
            cycles++;
        }
    }
}

// Función de partición para quickSort
int partition(int arr[], int low, int high, bool ascending, int
&cycles)
{
    int pivot = arr[high];
    int i = low - 1;

    for (int j = low; j <= high - 1; ++j)
    {
        if ((ascending && arr[j] < pivot) || (!ascending && arr[j] >
pivot))
        {
            ++i;
            swap(arr[i], arr[j]);
        }
        cycles++;
    }

    swap(arr[i + 1], arr[high]);
    return i + 1;
}

// Implementación de la función de ordenamiento quickSort
void quickSort(int arr[], int low, int high, bool ascending, int
&cycles)
{
    if (low < high)
    {
        int pi = partition(arr, low, high, ascending, cycles);

        quickSort(arr, low, pi - 1, ascending, cycles);
        quickSort(arr, pi + 1, high, ascending, cycles);
    }
}

```

```

// Función de fusión para mergeSort
void merge(int arr[], int low, int middle, int high, bool ascending,
int &cycles)
{
    int n1 = middle - low + 1;
    int n2 = high - middle;

    int left[n1], right[n2];

    for (int i = 0; i < n1; ++i)
    {
        left[i] = arr[low + i];
    }
    for (int j = 0; j < n2; ++j)
    {
        right[j] = arr[middle + 1 + j];
    }

    int i = 0;
    int j = 0;
    int k = low;

    while (i < n1 && j < n2)
    {
        if ((ascending && left[i] <= right[j]) || (!ascending &&
left[i] >= right[j]))
        {
            arr[k] = left[i];
            ++i;
        }
        else
        {
            arr[k] = right[j];
            ++j;
        }
        ++k;
        cycles++;
    }

    while (i < n1)
    {
        arr[k] = left[i];
        ++i;
        ++k;
        cycles++;
    }

    while (j < n2)
    {
        arr[k] = right[j];
        ++j;
        ++k;
        cycles++;
    }
}

// Implementación de la función de ordenamiento mergeSort

```

```

void mergeSort(int arr[], int low, int high, bool ascending, int
&cycles)
{
    if (low < high)
    {
        int middle = low + (high - low) / 2;

        mergeSort(arr, low, middle, ascending, cycles);
        mergeSort(arr, middle + 1, high, ascending, cycles);

        merge(arr, low, middle, high, ascending, cycles);
    }
}

// Función principal

int main()
{
    system("Color 4F");
    //color de la consola
    //system("Color F0");
    // Tamaño del array
    int size;
    cout <<"|                                MENU DE ORDENACIONES
|"<<endl;

    cout <<"Ingrese el tama"<<char(164)<<"o del array: "<<endl<<endl;
    cin >> size;

    if (size <= 0)
    {
        cout << "Tamaño no válido. Saliendo del programa.\n";
        return 1;
    }

    // Crear el array
    int arr[size];

    // Llenar el array con números aleatorios
    fillArrayRandom(arr, size);

    // Contador de ciclos
    int cycles = 0;

    // Menú de selección ascendente o descendente
    int choice;
    bool ascending;
    cout << "Seleccione el orden:"<<endl;
    cout << "1. Ascendente\n2. Descendente\n";
    cout << "Ingrese el numero correspondiente al orden: ";
    cin >> choice;
    ascending = (choice == 1);

    // Imprimir el array original
    cout << "Array original: ";
    printArray(arr, size);
    cout << endl;

```



```

// Ejecutar y mostrar resultados de cada método de ordenamiento
cout << "Resultados de los metodos de ordenamiento:\n";
cout << left << setw(15) << "Metodo" << setw(26) << "Array
ordenado" << setw(0) << "Ciclos\n";
cout << "-----\n";

cout << left << setw(15) << "Bubble Sort" << " ==> ";
int arr1[size];
copy(arr, arr + size, arr1);
bubbleSort(arr1, size, ascending, cycles);
printArray(arr1, size);
cout << setw(9) << "===> " << cycles << endl;
int grafico1=cycles;
cycles = 0;

cout << left << setw(15) << "Insertion Sort" << " ==> ";
int arr2[size];
copy(arr, arr + size, arr2);
insertionSort(arr2, size, ascending, cycles);
printArray(arr2, size);
cout << setw(9) << "===> " << cycles << endl;
int grafico2=cycles;
cycles = 0;

cout << left << setw(15) << "Selection Sort" << " ==> ";
int arr3[size];
copy(arr, arr + size, arr3);
selectionSort(arr3, size, ascending, cycles);
printArray(arr3, size);
cout << setw(9) << "===>" << cycles << endl;
int grafico3=cycles;
cycles = 0;

cout << left << setw(15) << "Counting Sort" << " ==> ";
int arr4[size];
copy(arr, arr + size, arr4);
countingSort(arr4, size, ascending, cycles);
printArray(arr4, size);
cout << setw(9) << "===>" << cycles << endl;
int grafico4=cycles;
cycles = 0;

cout << left << setw(15) << "Quick Sort" << " ==> ";
int arr5[size];
copy(arr, arr + size, arr5);
quickSort(arr5, 0, size - 1, ascending, cycles);
printArray(arr5, size);
cout << setw(9) << "===>" << cycles << endl;
int grafico5=cycles;
cycles = 0;

cout << left << setw(15) << "Merge Sort" << " ==> ";
int arr6[size];
copy(arr, arr + size, arr6);
mergeSort(arr6, 0, size - 1, ascending, cycles);

```

```

printArray(arr6, size);
cout << setw(9) <<"==="<<"cycles << endl;
int grafico6=cycles;

//GRAFIQUITO
cout <<endl;
cout<<"GRAFICO";
cout <<endl;
cout<<endl;
cout << left << setw(15) << "Bubble Sort" << " ==>
";drawBar(grafico1);
cout << left << setw(15) << "Insertion Sort" << " ==>
";drawBar(grafico2);
cout << left << setw(15) << "Selection Sort" << " ==>
";drawBar(grafico3);
cout << left << setw(15) << "Counting Sort" << " ==>
";drawBar(grafico4);
cout << left << setw(15) << "Quick Sort" << " ==>
";drawBar(grafico5);
cout << left << setw(15) << "Merge Sort" << " ==>
";drawBar(grafico6);

/*
for (int i = 0; i < cantidadNumeros; ++i)
{
    cin >> numeros[i];
}
*/
// Encontrar el menor número en el arreglo
cout<<endl;
int
cantidadNumeros[]={grafico1,grafico2,grafico3,grafico4,grafico5,grafico6};
int menor = cantidadNumeros[0]; // Asumir que el primer número es
el menor

for (int i = 0; i < 6; ++i)
{
    if (cantidadNumeros[i] < menor)
    {
        menor = cantidadNumeros[i];
    }
}

if(menor==grafico1)
{
    cout<<"Bubble Sort es el mejor metodo sus ciclos son:
"<<grafico1<<endl;
}
if(menor==grafico2)
{
    cout<<"Insertion Sort es el mejor metodo sus ciclos son:
"<<grafico2<<endl;
}
if(menor==grafico3)
{

```

```
        cout<<"Selection Sortes el mejor metodo sus ciclos son:
"<<grafico3<<endl;
    }
    if(menor==grafico4)
    {
        cout<<"Counting Sort es el mejor metodo sus ciclos son:
"<<grafico4<<endl;
    }
    if(menor==grafico5)
    {
        cout<<"Quick Sort es el mejor metodo sus ciclos son:
"<<grafico5<<endl;
    }
    if(menor==grafico6)
    {
        cout<<"Merge Sort es el mejor metodo sus ciclos son:
"<<grafico6<<endl;
    }

    return 0;
}
```