

CUATRIMESTRE 2°



ESTRUCTURA DE DATOS

Profesor: Valdez Hernandez Juanh
Augusto

Presentado por: Khevin Cruz Hernández

Matricula: 2331123331

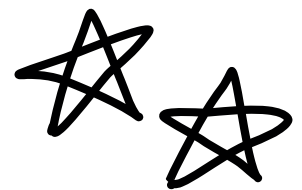
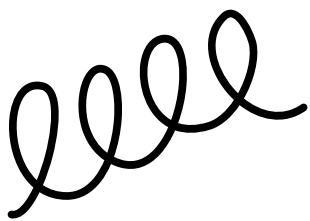
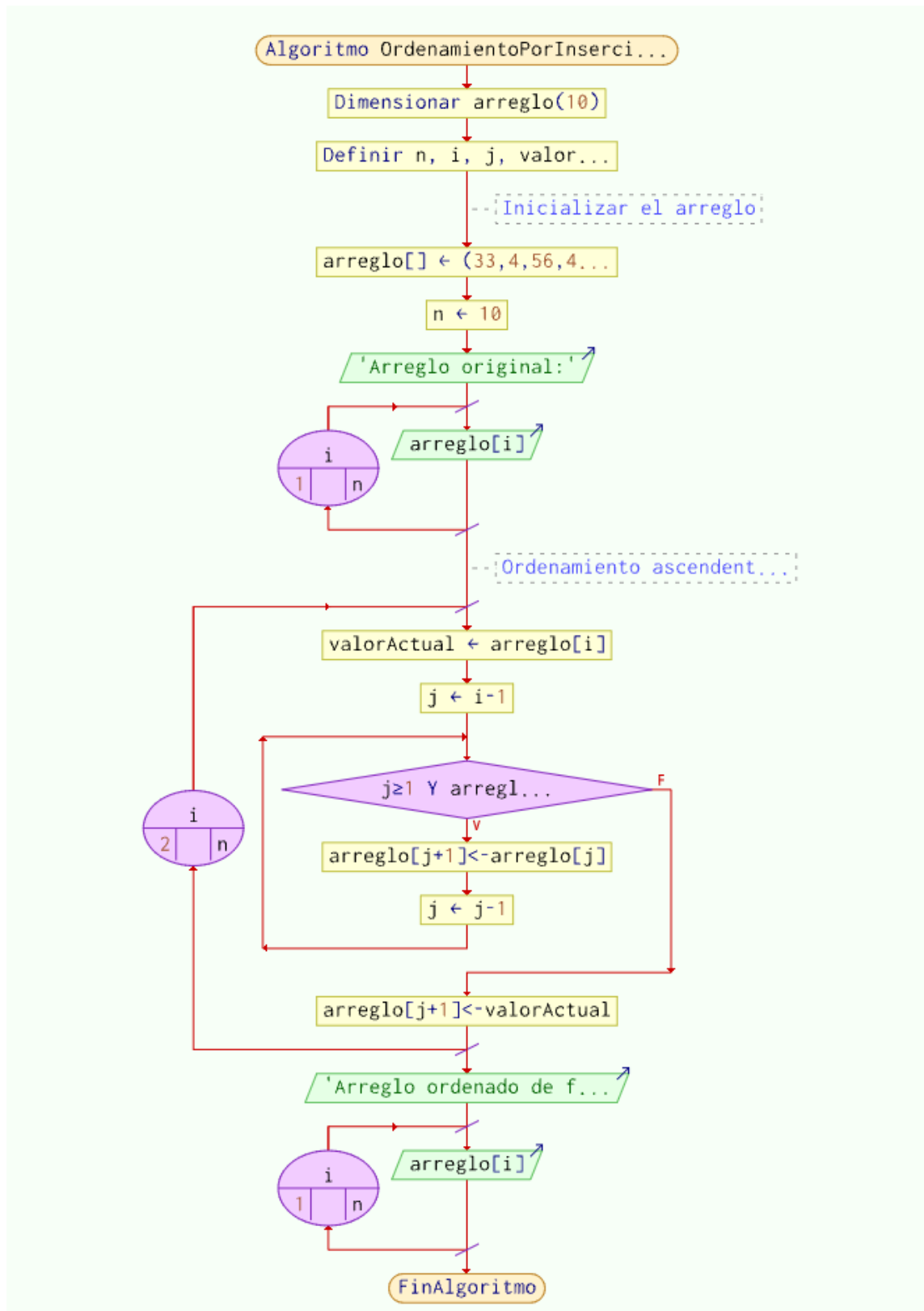
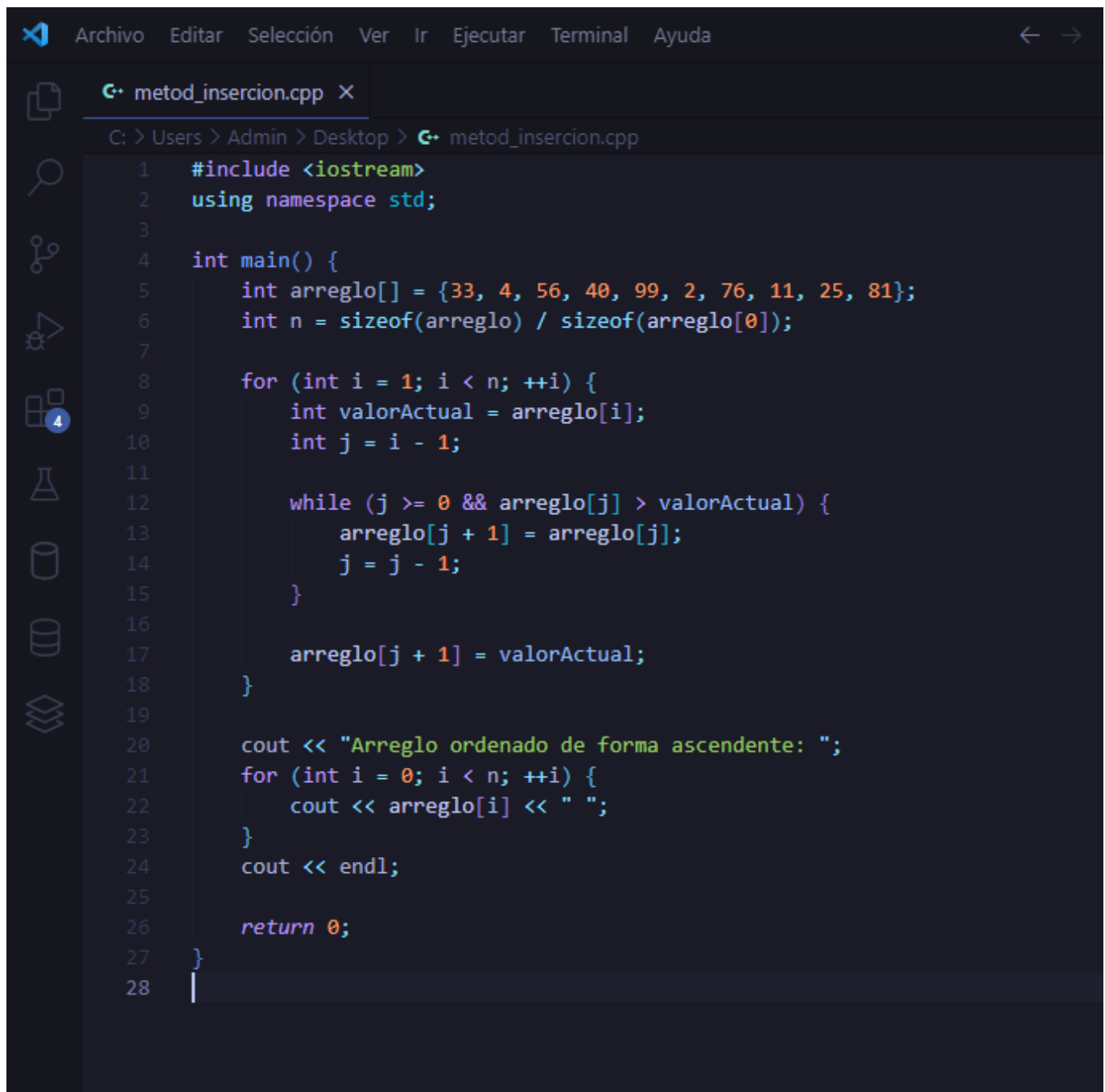


Diagrama de flujo



Código



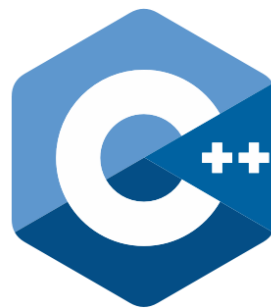
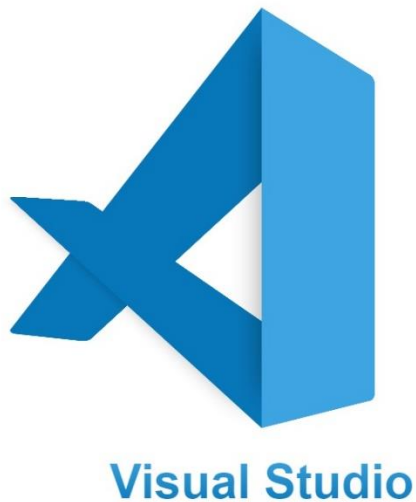
The image shows a screenshot of a C++ code editor with a dark theme. The editor has a menu bar at the top with options: Archivo, Editar, Selección, Ver, Ir, Ejecutar, Terminal, and Ayuda. Below the menu bar is a tab labeled 'metod_insercion.cpp'. The file path is shown as 'C:\> Users > Admin > Desktop > metod_insercion.cpp'. The code is written in C++ and implements an insertion sort algorithm. It includes the `<iostream>` header and uses the `std` namespace. The `main` function starts with an array `arreglo` containing the values {33, 4, 56, 40, 99, 2, 76, 11, 25, 81}. It calculates the size of the array as `n`. A `for` loop iterates from `i = 1` to `n - 1`. Inside the loop, it assigns `valorActual` to `arreglo[i]` and sets `j = i - 1`. A `while` loop shifts elements greater than `valorActual` one position to the right. Finally, it inserts `valorActual` at position `j + 1`. After the loop, it prints the sorted array and returns 0.

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int arreglo[] = {33, 4, 56, 40, 99, 2, 76, 11, 25, 81};
6      int n = sizeof(arreglo) / sizeof(arreglo[0]);
7
8      for (int i = 1; i < n; ++i) {
9          int valorActual = arreglo[i];
10         int j = i - 1;
11
12         while (j >= 0 && arreglo[j] > valorActual) {
13             arreglo[j + 1] = arreglo[j];
14             j = j - 1;
15         }
16
17         arreglo[j + 1] = valorActual;
18     }
19
20     cout << "Arreglo ordenado de forma ascendente: ";
21     for (int i = 0; i < n; ++i) {
22         cout << arreglo[i] << " ";
23     }
24     cout << endl;
25
26     return 0;
27 }
28
```

Implementación

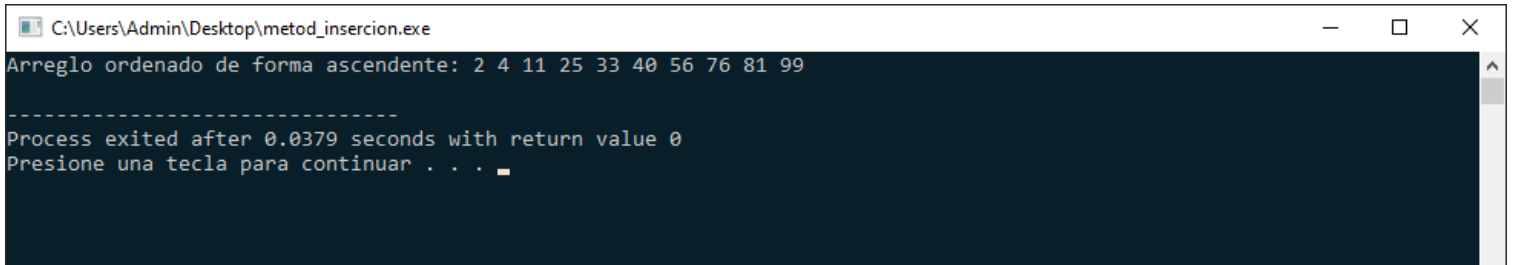
Para poner en marcha el código se busca un entorno de desarrollo integrado (IDE) que admita C++, se crea un archivo nuevo y copia el código del algoritmo de método de ordenamiento por inserción en el archivo. Una vez teniendo el código se compila y si no hay errores de compilación, se crea un archivo ejecutable, deberíamos ver la salida del programa, que incluirá el arreglo original y el arreglo ordenado.

La implementación del código consiste en ordenar un arreglo de 10 valores enteros de forma ascendente utilizando el método de inserción. Comienza con la inicialización del arreglo con valores predefinidos, seguido de la visualización del arreglo original. Luego, se ejecuta el algoritmo de ordenamiento por inserción, donde cada elemento se compara con los elementos anteriores y se coloca en la posición correcta. Finalmente, se muestra el arreglo ordenado de forma ascendente. Este enfoque proporciona una solución clara y eficiente para organizar los elementos del arreglo de manera sistemática y comprensible.



Resultados

El resultado arrojado del algoritmo es un arreglo ordenado de forma ascendente usando el método de ordenamiento por insercion:



```
C:\Users\Admin\Desktop\metod_insercion.exe
Arreglo ordenado de forma ascendente: 2 4 11 25 33 40 56 76 81 99
-----
Process exited after 0.0379 seconds with return value 0
Presione una tecla para continuar . . .
```

Conclusión

En conclusión, la implementación del algoritmo de ordenamiento por inserción para ordenar un arreglo de 10 valores enteros de forma ascendente resulta en una solución efectiva y comprensible. Este método proporciona una manera sencilla y eficiente de organizar los elementos del arreglo, asegurando que estén dispuestos en el orden adecuado. La claridad y la facilidad de comprensión de este algoritmo lo hacen adecuado para aplicaciones donde se requiera ordenar conjuntos de datos pequeños de manera rápida y precisa. Además, el código proporciona una base sólida para comprender los principios fundamentales de los algoritmos de ordenamiento y su aplicación en la práctica de la programación.