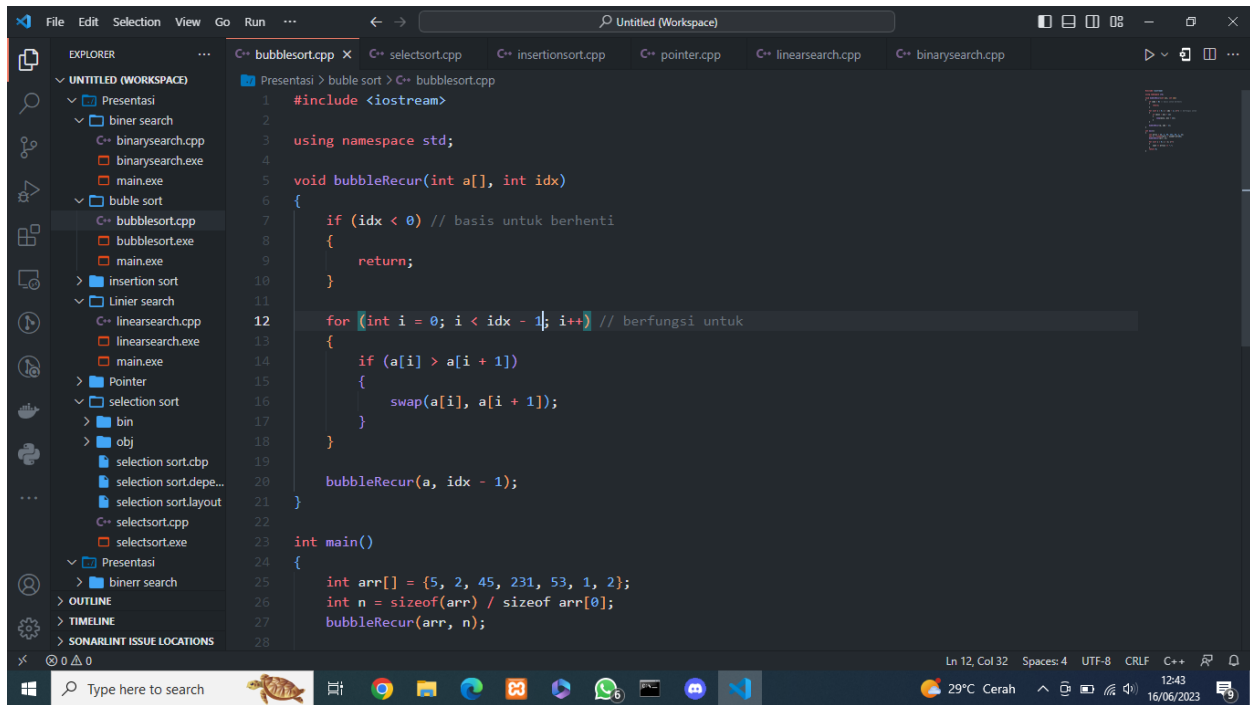


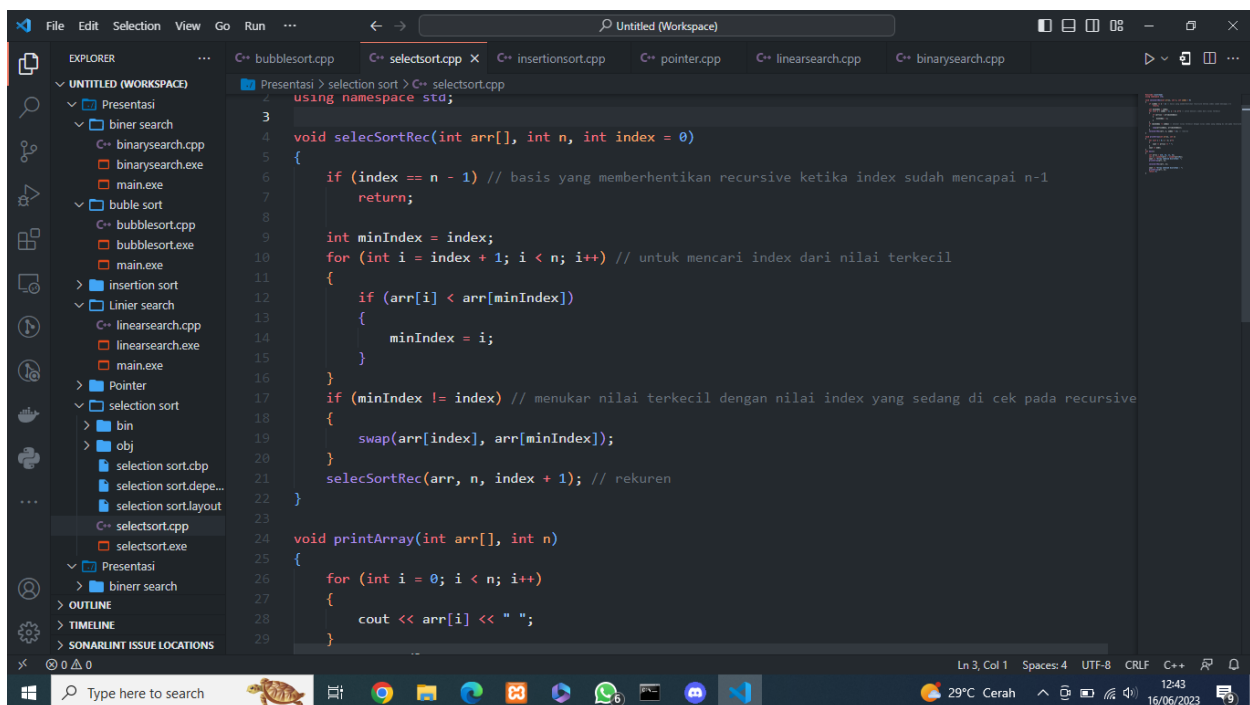
Bubble



The screenshot shows the Visual Studio Code editor with a C++ project named 'Presentasi'. The file explorer on the left shows a directory structure with folders for 'biner search', 'bubble sort', 'insertion sort', 'Linier search', 'Pointer', 'selection sort', and 'bin'. The 'bubble sort' folder is expanded, showing 'bubblesort.cpp', 'bubblesort.exe', and 'main.exe'. The main editor displays the code for 'bubblesort.cpp'.

```
1 #include <iostream>
2
3 using namespace std;
4
5 void bubbleRecur(int a[], int idx)
6 {
7     if (idx < 0) // basis untuk berhenti
8     {
9         return;
10    }
11
12    for (int i = 0; i < idx - 1; i++) // berfungsi untuk
13    {
14        if (a[i] > a[i + 1])
15        {
16            swap(a[i], a[i + 1]);
17        }
18    }
19
20    bubbleRecur(a, idx - 1);
21 }
22
23 int main()
24 {
25     int arr[] = {5, 2, 45, 231, 53, 1, 2};
26     int n = sizeof(arr) / sizeof arr[0];
27     bubbleRecur(arr, n);
28 }
```

Selection



The screenshot shows the Visual Studio Code editor with the same C++ project. The file explorer on the left shows the 'selection sort' folder expanded, showing 'selection sort.cpp', 'selection sort.exe', and 'main.exe'. The main editor displays the code for 'selection sort.cpp'.

```
1 #include <iostream>
2 using namespace std;
3
4 void selecSortRec(int arr[], int n, int index = 0)
5 {
6     if (index == n - 1) // basis yang memberhentikan recursive ketika index sudah mencapai n-1
7     {
8         return;
9     }
10
11     int minIndex = index;
12     for (int i = index + 1; i < n; i++) // untuk mencari index dari nilai terkecil
13     {
14         if (arr[i] < arr[minIndex])
15         {
16             minIndex = i;
17         }
18     }
19
20     if (minIndex != index) // menukar nilai terkecil dengan nilai index yang sedang di cek pada recursive
21     {
22         swap(arr[index], arr[minIndex]);
23     }
24
25     selecSortRec(arr, n, index + 1); // rekuren
26 }
27
28 void printArray(int arr[], int n)
29 {
30     for (int i = 0; i < n; i++)
31     {
32         cout << arr[i] << " ";
33     }
34 }
```

Insertion

The screenshot shows the Visual Studio Code editor with a C++ project named 'insertionsort.cpp'. The code implements the insertion sort algorithm. The Explorer panel on the left shows the project structure with folders for 'biner search', 'bubble sort', 'insertion sort', 'linier search', and 'pointer'. The main editor displays the following code:

```
1 #include <iostream>
2 using namespace std;
3 void insertion(int arr[], int n)
4 {
5     // basis untuk berhenti
6     // Jika besar array kurang dari sama dengan 1
7     if (n <= 1)
8     {
9         return;
10    }
11    // Rekuren
12    insertion(arr, n - 1); // pengurangan array ketika 1 siklus perulangan sorting selesai
13    int last = arr[n - 1]; // nilai terakhir di dalam array
14    int beforelast = n - 2; // nilai sebelum nilai akhir di array
15    // Sorting
16    // Memindahkan nilai yang lebih besar dari last ke posisi yang lebih tinggi di array
17    while (beforelast >= 0 && arr[beforelast] > last)
18    { // perbandingan nilai sebelum akhir dengan nilai paling akhir
19        // jika nilai sebelum akhir lebih besar dari nilai last, maka:
20        arr[beforelast + 1] = arr[beforelast]; // pergantian nilai last
21        beforelast--; // pergantian index beforelast sebelum perulangan sorting samp
22    }
23    arr[beforelast + 1] = last; // Perulangan dengan nilai last yang berbeda
24 }
25
26 int main()
27 { // 0 1 2 3 4 5 6
28     int arr[] = {4, 3, 7, 9, 6, 5, 8};
```

Pointer

The screenshot shows the Visual Studio Code editor with a C++ project named 'pointer.cpp'. The code calculates the size of a pointer for different levels of recursion. The Explorer panel on the left shows the project structure with folders for 'biner search', 'bubble sort', 'insertion sort', 'linier search', and 'pointer'. The main editor displays the following code:

```
1 #include <iostream>
2 using namespace std;
3
4 int calculatePointerSize(int level)
5 {
6     if (level == 0)
7     {
8         void* ptr = nullptr;
9         return sizeof(ptr);
10    }
11
12    return calculatePointerSize(level - 1);
13 }
14
15 int main()
16 {
17     int level = 5;
18     int pointerSize = calculatePointerSize(level);
19
20     cout << "Ukuran pointer pada level " << level << ": " << pointerSize << " byte" << endl;
21     return 0;
22 }
23
24 +++++
```

Linear

The screenshot shows the Visual Studio Code editor with a C++ project named 'Presentasi'. The Explorer panel on the left shows the file structure, including folders for 'merge sort', 'linier search', 'Pointer', 'selection sort', and 'bubble sort'. The active file is 'linearsearch.cpp' in the 'linier search' folder. The code implements a recursive linear search function and a main function that tests it with an array [12, 34, 54, 2, 3].

```
1 #include <iostream>
2 using namespace std;
3
4 // fungsi rekursif pencarian linear
5
6 int linearSearchRecursive(int arr[], int n, int key, int index)
7 {
8     if (index >= n) //diuji sampai index n
9     {
10         return -1;
11     }
12     if (arr[index] == key)
13     {
14         return index;
15     }
16     return linearSearchRecursive(arr, n, key, index + 1);
17 }
18
19 int main()
20 {
21     int arr[] = {12, 34, 54, 2, 3};
22     int n = sizeof(arr) / sizeof(arr[0]);
23     int key = 54;
24
25     int result = linearSearchRecursive(arr, n, key, 0);
26     if (result == -1)
27     {
28         cout << "Elemen tidak ditemukan dalam array." << endl;
```

Binary

The screenshot shows the Visual Studio Code editor with the same C++ project, but now the active file is 'binarysearch.cpp' in the 'biner search' folder. The code implements a recursive binary search function and a main function. The search range is defined by 'left' and 'right' indices, and the middle element is calculated as $\text{mid} = \text{left} + (\text{right} - \text{left}) / 2$.

```
1 #include <iostream>
2 using namespace std;
3
4 // Fungsi rekursif untuk pencarian biner
5
6 int binarySearchRecursive(int arr[], int left, int right, int key)
7 {
8     if (right >= left)
9     {
10         // basis untuk berhenti jika array kosong dan memberhentikan
11         // menentukan nilai tengah sebagai acuan untuk membagi array
12         int mid = left + (right - left) / 2;
13
14         if (arr[mid] == key)
15         {
16             // basis untuk berhenti jika nilai yang dicari ditemukan
17             return mid;
18         }
19
20         if (arr[mid] > key)
21         {
22             return binarySearchRecursive(arr, left, mid - 1, key);
23         }
24
25         return binarySearchRecursive(arr, mid + 1, right, key);
26     }
27     return -1;
28 }
29
30 int main()
31 {
```