

NAMA : WILI CAHYADI

NPM : 0619103001

KELAS A REG B1 TEKNIK INFORMATIKA

TUGAS ALPRO ( SORTING FILE )

## 1. Menjalankan Contoh – contoh program di Modul

➤ // C program implementing Selection Sort

The image shows a C++ IDE with the following content:

```
1 // C program implementing Selection Sort
2 #include <stdio.h>
3 // function to swap elements of the given index values
4 void swap(int arr[], int firstIndex, int secondIndex) {
5     int temp;
6     temp = arr[firstIndex];
7     arr[firstIndex] = arr[secondIndex];
8     arr[secondIndex] = temp;
9 }
10 // function to look for smallest element in the given subarray
11 int indexOfMinimum(int arr[], int startIndex, int n) {
12     int minIndex = startIndex;
13     for(int i = startIndex + 1; i < n; i++) {
14         if(arr[i] < arr[minIndex]) {
15             minIndex = i;
16         }
17     }
18     return minIndex;
19 }
20 void selectionSort(int arr[], int n) {
21     for(int i = 0; i < n; i++) {
22         int index = indexOfMinimum(arr, i, n);
23         swap(arr, i, index);
24     }
25 }
26 void printArray(int arr[], int size) {
27     int i;
28     for(i = 0; i < size; i++) {
29         printf("%d ", arr[i]);
30     }
31     printf("\n");
32 }
33 int main() {
34     int arr[] = {46, 52, 21, 22, 11};
35     int n = sizeof(arr)/sizeof(arr[0]);
36     selectionSort(arr, n);
37     printf("Sorted array: \n");
38     printArray(arr, n);
39     return 0;
40 }
```

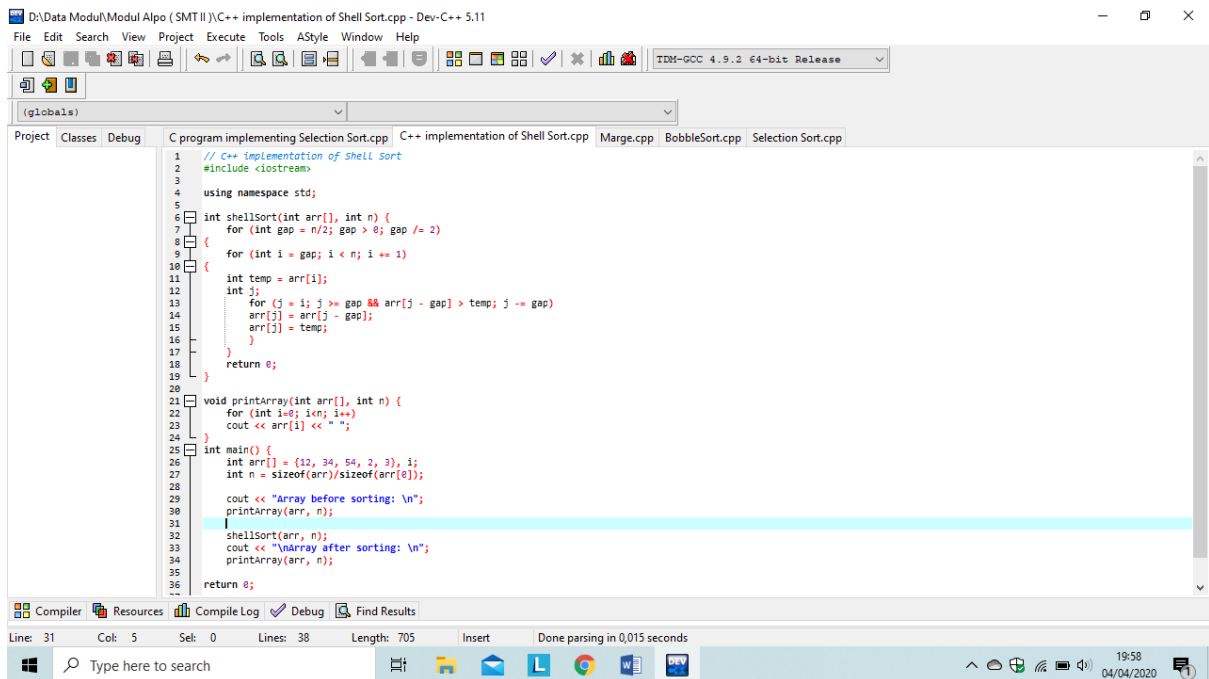
Output:

```
Sorted array:
11 21 22 46 52
Process exited after 0.07399 seconds with return value 0
Press any key to continue . . .
```

Compiler Output:

```
- Errors: 0
- Warnings: 0
- Output Filename: D:\Data Modul\Modul Alpo ( SMT II )\C program implementing Selection Sort.exe
- Output Size: 129,9560546875 KiB
- Compilation Time: 0,42s
```

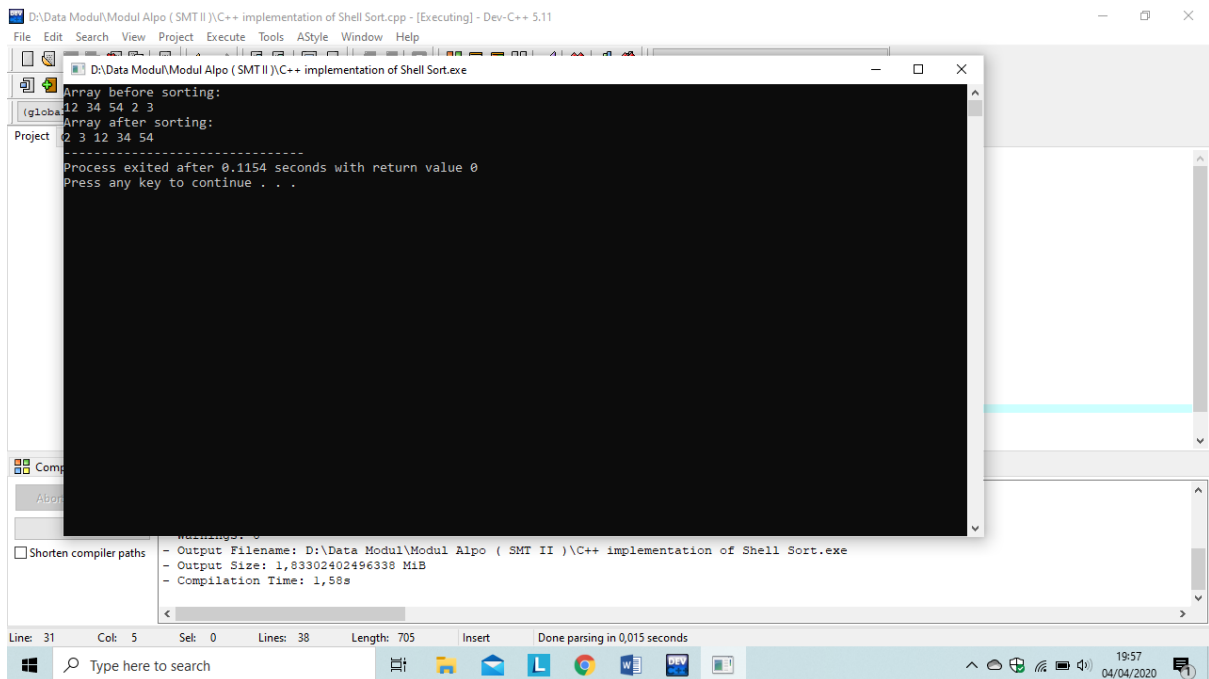
## ➤ // C++ implementation of Shell Sort



The screenshot shows the Dev-C++ IDE with a C++ program implementing Shell Sort. The code is as follows:

```
1 // C++ implementation of Shell Sort
2 #include <iostream>
3
4 using namespace std;
5
6 int shellSort(int arr[], int n) {
7     for (int gap = n/2; gap > 0; gap /= 2)
8     {
9         for (int i = gap; i < n; i += 1)
10        {
11            int temp = arr[i];
12            int j;
13            for (j = i; j >= gap && arr[j - gap] > temp; j -= gap)
14                arr[j] = arr[j - gap];
15            arr[j] = temp;
16        }
17    }
18    return 0;
19 }
20
21 void printArray(int arr[], int n) {
22     for (int i=0; i<n; i++)
23         cout << arr[i] << " ";
24 }
25
26 int main() {
27     int arr[] = {12, 34, 54, 2, 3};
28     int n = sizeof(arr)/sizeof(arr[0]);
29     cout << "Array before sorting: \n";
30     printArray(arr, n);
31     shellSort(arr, n);
32     cout << "\nArray after sorting: \n";
33     printArray(arr, n);
34     return 0;
35 }
```

The status bar at the bottom indicates: Line: 31, Col: 5, Sel: 0, Lines: 38, Length: 705, Insert, Done parsing in 0,015 seconds.

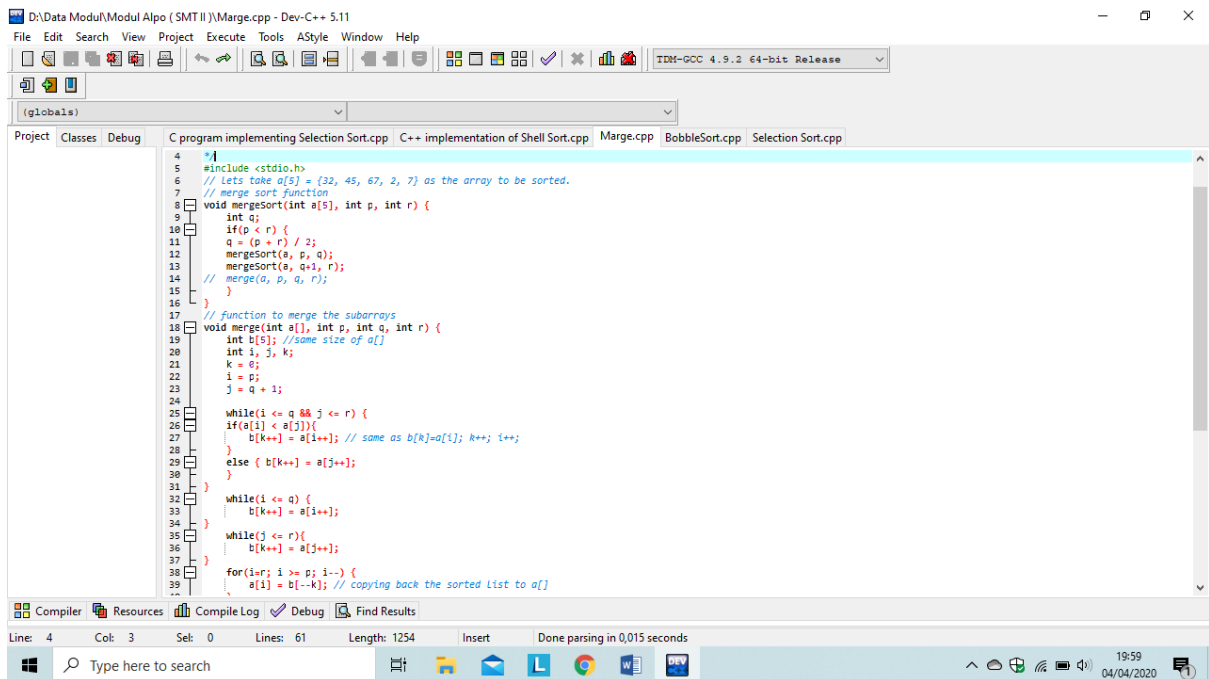


The screenshot shows the execution output of the Shell Sort program. The output is as follows:

```
Array before sorting:
12 34 54 2 3
Array after sorting:
2 3 12 34 54
-----
Process exited after 0.1154 seconds with return value 0
Press any key to continue . . .
```

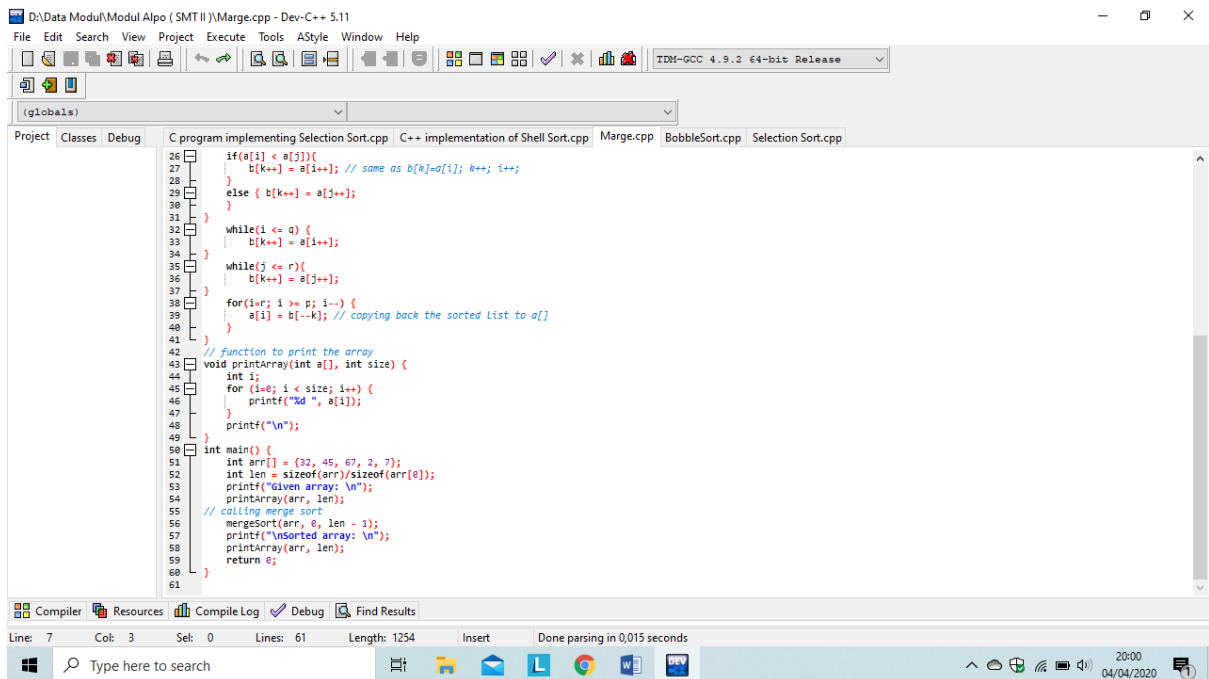
The status bar at the bottom indicates: Line: 31, Col: 5, Sel: 0, Lines: 38, Length: 705, Insert, Done parsing in 0,015 seconds.

## ➤ Merge



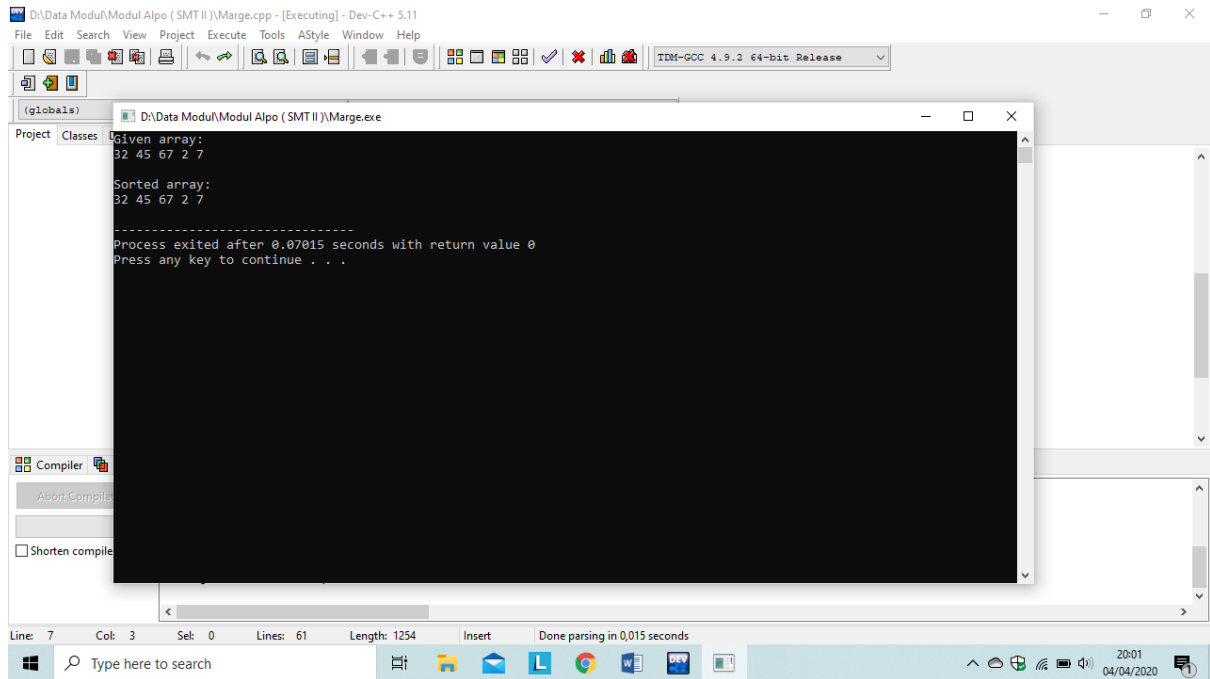
The screenshot shows a C++ IDE with the file `Merge.cpp` open. The code implements the merge function, which takes an array `a` and indices `p`, `q`, and `r`. It first divides the array into two halves and recursively sorts them. Then, it merges the two sorted halves back into the original array. The merge process involves creating a temporary array `b` and copying elements from `a` into `b` based on their relative values. Finally, the sorted elements are copied back into `a`.

```
4 #include <stdio.h>
5 // Lets take a[] = {32, 45, 67, 2, 7} as the array to be sorted.
6 // merge sort function
7 void mergeSort(int a[], int p, int r) {
8     int q;
9     if (p < r) {
10         q = (p + r) / 2;
11         mergeSort(a, p, q);
12         mergeSort(a, q+1, r);
13         // merge(a, p, q, r);
14     }
15 }
16 // function to merge the subarrays
17 void merge(int a[], int p, int q, int r) {
18     int b[s]; //same size of a[]
19     int i, j, k;
20     k = 0;
21     i = p;
22     j = q + 1;
23     while(i <= q && j <= r) {
24         if(a[i] < a[j]){
25             b[k++] = a[i++]; // same as b[k]=a[i]; k++; i++;
26         }
27         else { b[k++] = a[j++];
28             }
29     }
30     while(i <= q) {
31         b[k++] = a[i++];
32     }
33     while(j <= r){
34         b[k++] = a[j++];
35     }
36     for(i=r; i >= p; i--) {
37         a[i] = b[--k]; // copying back the sorted list to a[]
38     }
39 }
40
41 // function to print the array
42 void printArray(int a[], int size) {
43     int i;
44     for (i=0; i < size; i++) {
45         printf("%d ", a[i]);
46     }
47     printf("\n");
48 }
49
50 int main() {
51     int arr[] = {32, 45, 67, 2, 7};
52     int len = sizeof(arr)/sizeof(arr[0]);
53     printf("Given array: \n");
54     printArray(arr, len);
55     // calling merge sort
56     mergeSort(arr, 0, len - 1);
57     printf("\nsorted array: \n");
58     printArray(arr, len);
59     return 0;
60 }
```



The screenshot shows the same C++ IDE with the file `Merge.cpp` open. The code continues with the `mergeSort` function and the `main` function. The `main` function initializes an array `arr` with the values {32, 45, 67, 2, 7}, prints it, calls `mergeSort` to sort it, and then prints the sorted array.

```
26 if(a[i] < a[j]){
27     b[k++] = a[i++]; // same as b[k]=a[i]; k++; i++;
28 }
29 else { b[k++] = a[j++];
30 }
31 }
32 while(i <= q) {
33     b[k++] = a[i++];
34 }
35 while(j <= r){
36     b[k++] = a[j++];
37 }
38 for(i=r; i >= p; i--) {
39     a[i] = b[--k]; // copying back the sorted list to a[]
40 }
41 }
42 // function to print the array
43 void printArray(int a[], int size) {
44     int i;
45     for (i=0; i < size; i++) {
46         printf("%d ", a[i]);
47     }
48     printf("\n");
49 }
50
51 int main() {
52     int arr[] = {32, 45, 67, 2, 7};
53     int len = sizeof(arr)/sizeof(arr[0]);
54     printf("Given array: \n");
55     printArray(arr, len);
56     // calling merge sort
57     mergeSort(arr, 0, len - 1);
58     printf("\nsorted array: \n");
59     printArray(arr, len);
60     return 0;
61 }
```



## 2. Contoh Program Bubble Sort dan Selection Sort

### ➤ BubbleSort

```

#include <stdio.h>

void bubbleSort(int arr[], int n){
    int i, j, tmp;
    for(i = 0; i < n; i++){
        for(j=0; j < n-i-1; j++){
            if(arr[j] > arr[j+1]){
                tmp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = tmp;
            }
        }
    }
}

int main() {
    int array[10], n, i, j;
    printf("Masukkan banyak elemen: ");
    scanf("%d", &n);
    printf("Masukkan nilai: \n");
    for(i = 0; i < n; i++){
        scanf(" %d ", &array[i]);
    }
    bubbleSort(array, n);
    printf("Hasil pengurutan sebagai berikut:\n");
    for(i = 0; i < n; i++){
        printf("%d ", array[i]);
    }
    printf("\n");
}

```

➤ Selection Sort

```
#include <iostream>
#include <conio.h>
#include <stdio.h>
#include <iomanip>
#ifdef __cplusplus__
#include <cstdlib>
#else
#include <stdlib.h>
#endif

using namespace std;
int main() {
    int x[10];
    int i;
    int temp;
    int minindex;
    int j;

    if (system("CLS")) system("clear");
    cout << ">> Program Selection Sort << \n" << endl;
    cout << "masukkan nilai x :\n";
    for (i = 0; i<10; i++) {
        cout << "x[" << i << "] = "; cin >> x[i];
    }
    cout << "\n Data sebelum di sort :";
    for (i = 0; i<10; i++) {
        cout << setw(9) << x[i];
    }
    for (i = 0; i<10 - 1; i++) //perulangan iterasi
    {
        minindex = i;
        for (j = i + 1; j<10; j++) //perulangan membandingkan data
        {
            if (x[minindex]>x[j])
            {
                minindex = j;
            }
        }
        temp = x[i];
        x[i] = x[minindex];
        x[minindex] = temp;
    }
    cout << "\n Data setelah di sort :";
    for (i = 0; i<10; i++)
    {
        cout << setw(9) << x[i];
    }
}
```

```

    }

    getchar();

    cout << endl;

    system("pause");

}

```

### 3. Kekurangan dan Kelebihan metode Bubble Sort dan Selection Sort

#### ➤ Kelebihan Bubble Sort :

Metode Buble Sort merupakan metode yang paling simple  
Metode Buble Sort mudah dipahami algoritmanya

#### ➤ Kelemahan Bubble Sort:

Meskipun simpel metode Bubble sort merupakan metode pengurutan yang paling tidak efisien. *Kelemahan buble sort* adalah pada saat mengurutkan data yang sangat besar akan mengalami kelambatan luar biasa, atau dengan kata lain kinerja memburuk cukup signifikan ketika data yang diolah jika data cukup banyak. Kelemahan lain adalah jumlah pengulangan akan tetap sama jumlahnya walaupun data sesungguhnya sudah cukup terurut. Hal ini disebabkan setiap data dibandingkan dengan setiap data yang lain untuk menentukan posisinya.

#### ➤ Kelebihan Selection Sort :

- Algoritma ini sangat rapat dan mudah untuk diimplementasikan.
- Mempercepat pencarian
- Mudah menentukan data maksimum /minimum.
- Mudah menggabungkannya kembali. Kompleksitas selection sort relatif lebih kecil.

#### ➤ Kekurangan Selection Sort :

- Membutuhkan method tambahan
- Sulit untuk digabungkan kembali
- Perlu dihindari untuk penggunaan data lebih dari 1000 tabel, karena akan menyebabkan kompleksitas yang lebih tinggi dan kurang praktis