

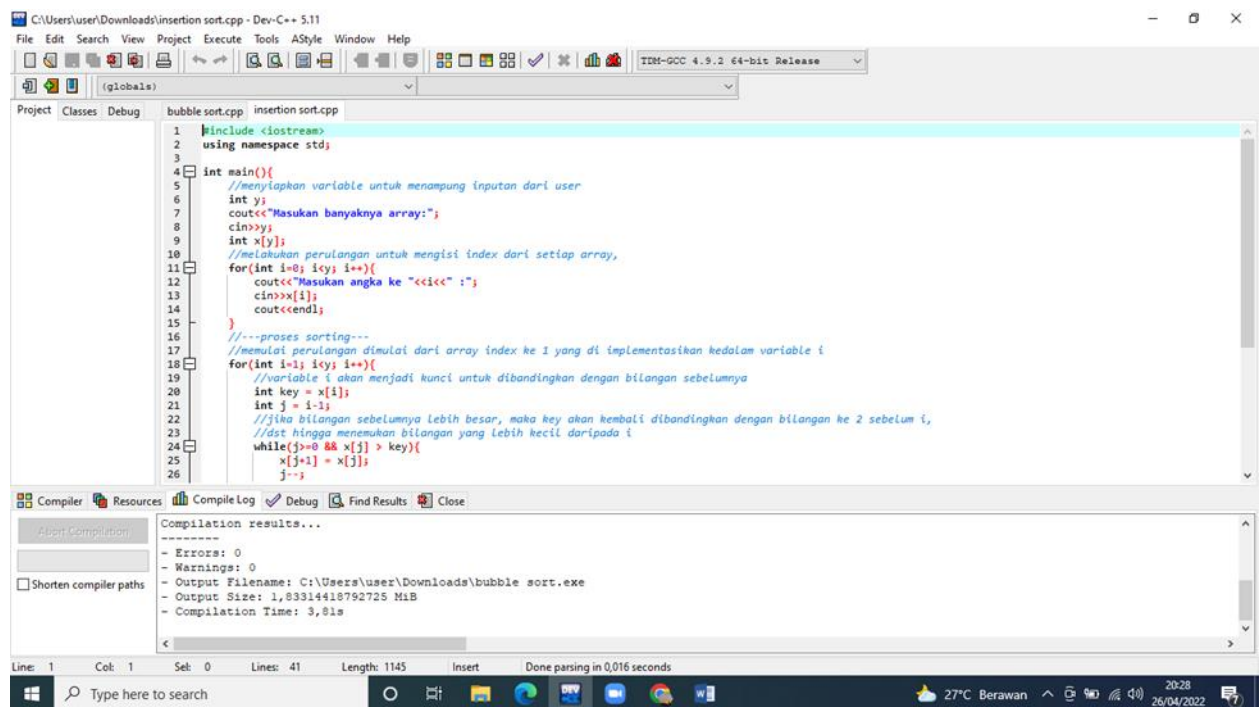
# LAPORAN KELOMPOK STUKTUR DATA

## NAMA KELOMPOK

Nila Gayatri	21091397066
Ahmed Nur Sidik	21091397038
Affandika Febrian Putra Yunanto	21091397030
Ahmad Donny Damanik	21091397008
Diego Athalla Samudero	21091397042

1. Sorting yang paling cepat dan jelaskan kenapa
2. Sorting yang paling lambat dan jelaskan kenapa

## Insertion sort



```
1 #include <iostream>
2 using namespace std;
3
4 int main(){
5     //menyiapkan variable untuk menampung inputan dari user
6     int y;
7     cout<<"Masukan banyaknya array:";
8     cin>>y;
9     int x[y];
10    //melakukan perulangan untuk mengist index dari setiap array,
11    for(int i=0; i<y; i++){
12        cout<<"Masukan angka ke "<<i<<" :";
13        cin>>x[i];
14        cout<<endl;
15    }
16    //---proses sorting---
17    //memulai perulangan dimulai dari array index ke 1 yang di implementasikan kedalam variable i
18    for(int i=1; i<y; i++){
19        //variable i akan menjadi kunci untuk dibandingkan dengan bilangan sebelumnya
20        int key = x[i];
21        int j = i-1;
22        //jika bilangan sebelumnya lebih besar, maka key akan kembali dibandingkan dengan bilangan ke 2 sebelum i,
23        //dst hingga menemukan bilangan yang lebih kecil daripada i
24        while(j>0 && x[j] > key){
25            x[j+1] = x[j];
26            j--;
```

Compilation results...

- Errors: 0

- Warnings: 0

- Output Filename: C:\Users\user\Downloads\bubble sort.exe

- Output Size: 1,83314418792725 MiB

- Compilation Time: 3,81s

Dimana Data akan dicek satu per satu untuk mulai dari yang kedua dengan sampai akhir dimana apabila ditemukan data yang lebih kecil dan daripada data sebelumnya, maka data tersebut akan pastikan di sisikan pada posisi yang akan sesuai, dan akan lebih mudah apabila pengurutan kartu.

## Selection sort

The screenshot shows a C++ IDE with the following code in `void 1.cpp`:

```
1 #include <iostream>
2 using namespace std;
3 void swapping(int &a, int &b){ //menukar isi dari a dan b
4     int temp;
5     temp = a;
6     a = b;
7     b = temp;
8 }
9 void display(int *array, int size){
10     for(int i = 0; i < size; i++){
11         cout << array[i] << " ";
12         cout << endl;
13     }
14 }
15 void selectionSort(int *array, int size){
16     int i, j, imin;
17     for(i = 0; i < size-1; i++){
18         imin = i; //mendapatkan indeks data minimum
19         for(j = i+1; j < size; j++){
20             if(array[j] < array[imin])
21                 imin = j; //menempatkan di posisi yang benar
22         }
23         swap(array[i], array[imin]);
24     }
25 }
26 int main(){
27     int n;
```

The compilation results show:

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\user\Downloads\bubble sort.exe
- Output Size: 1,83314418792725 MiB
- Compilation Time: 3,81s

Data tersebut dicek satu per satu mulai dari yang kedua sampai dengan yang terakhir. Apabila ditemukan data yang lebih kecil daripada data sebelumnya, maka data tersebut akan disisipkan pada posisi yang sesuai. Kemudian Akan lebih mudah apabila membayangkan pengurutan kartu.

## Radix sort

The screenshot shows a C++ IDE with the following code in `main.cpp`:

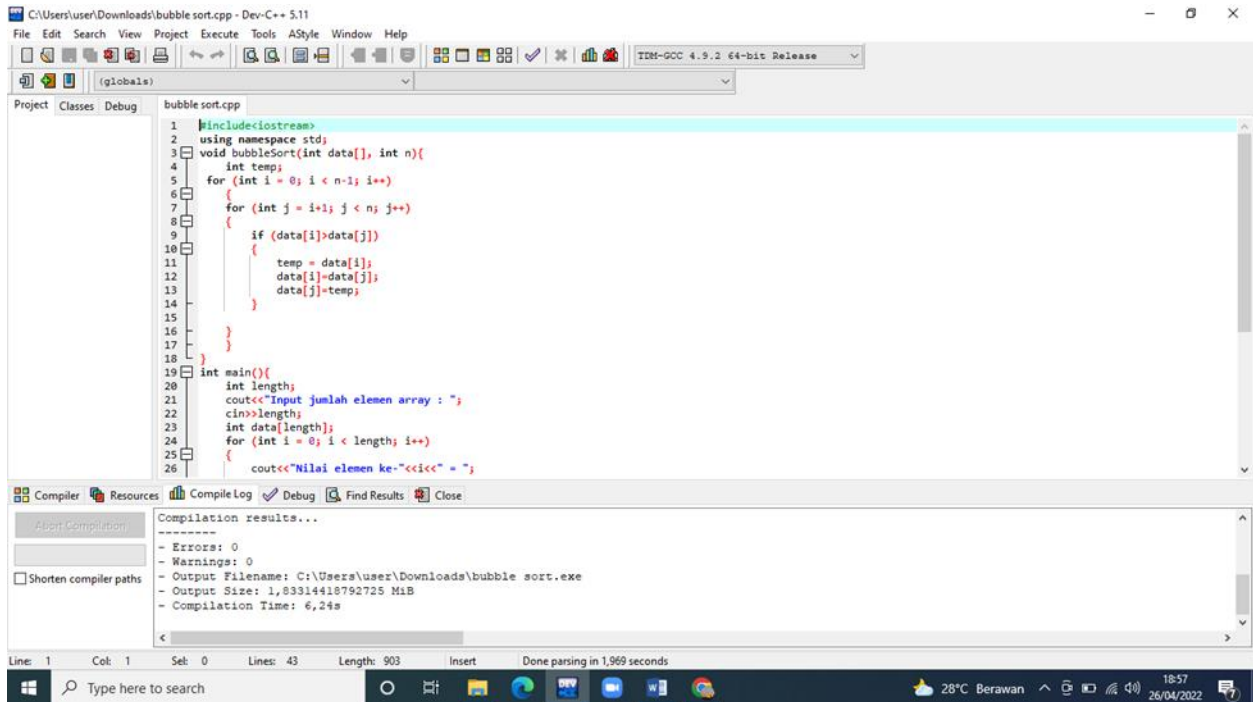
```
1 #include <iostream>
2 using namespace std;
3 void countSort(int nilai[], int inputan, int k){
4     int output[inputan];
5     int i, NILAI[inputan] = {0};
6     for (i = 0; i < inputan; i++){
7         NILAI[ (nilai[i]/k)%10 ]++;
8     }
9     for (i = 1; i < 10; i++){
10         NILAI[i] += NILAI[i - 1];
11     }
12     for (i = inputan - 1; i >= 0; i--){
13         output[ NILAI[ (nilai[i]/k)%10 ] - 1 ] = nilai[i];
14         NILAI[ (nilai[i]/k)%10 ]--;
15     }
16     for (i = 0; i < inputan; i++){
17         nilai[i] = output[i];
18     }
19 }
20 void radixSort(int nilai[], int inputan){
21     int z = nilai[0];
22     for (int i = 1; i < inputan; i++){
23         if (z < nilai[i]){
24             z = nilai[i];
25         }
26     }
```

The compilation results show:

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\user\Downloads\bubble sort.exe
- Output Size: 1,83314418792725 MiB
- Compilation Time: 3,81s

**Radix sort** adalah merupakan metode pengurutan yang dengan tidak ada perbandingan antara pengurutan dimana dalam prosesnya tidak melakukan perbandingan antar data

## Bubble sort



The screenshot shows a C++ IDE with the file 'bubble sort.cpp'. The code implements a bubble sort algorithm. It includes `<iostream>` and uses the `std` namespace. The `bubbleSort` function takes an array of integers and its size `n`. It uses two nested loops: the outer loop iterates from `i = 0` to `n-1`, and the inner loop iterates from `j = i+1` to `n`. Inside the inner loop, it compares `data[i]` and `data[j]`. If `data[i] > data[j]`, it swaps them using a temporary variable `temp`. The `main` function prompts the user to input the number of elements in the array, reads the array, and prints each element.

```
1 #include<iostream>
2 using namespace std;
3 void bubbleSort(int data[], int n){
4     int temp;
5     for (int i = 0; i < n-1; i++)
6     {
7         for (int j = i+1; j < n; j++)
8         {
9             if (data[i]>data[j])
10            {
11                temp = data[i];
12                data[i]=data[j];
13                data[j]=temp;
14            }
15        }
16    }
17 }
18
19 int main(){
20     int length;
21     cout<<"Input jumlah elemen array : ";
22     cin>>length;
23     int data[length];
24     for (int i = 0; i < length; i++)
25     {
26         cout<<"Nilai elemen ke-"<<i<<" = ";
```

The compilation results show 0 errors and 0 warnings. The output filename is `C:\Users\user\Downloads\bubble sort.exe`, the output size is 1,833,144,187,927,25 MiB, and the compilation time is 6.24s.

Bubble sort merupakan metode algoritma dengan cara melakukan penukaran data yang dengan di sebelah seara terus menerus hingga sampai bisa untuk di pastikan dalam satu iterasi dengan tertentu dan tidak lagi ada perubahan dan ketika ada perubahan berate ada data yang sudah terurut dan bubble sort disebut dengan penguruta masing masing kuci akan berbeda beda dalam memproses tempat posisinya.

## Merge sort

The screenshot shows a C++ IDE with a project named 'merge sortku'. The main file, 'merge sortku.cpp', contains the following code:

```
1 #include <iostream>
2 using namespace std;
3
4 void merge(int arr[], int l, int m, int r)
5 {
6     int a, b, c;
7     int n1 = m - l + 1;
8     int n2 = r - m;
9
10    int L[n1], R[n2];
11
12    for (a = 0; a < n1; a++)
13        L[a] = arr[l + a];
14    for (b = 0; b < n2; b++)
15        R[b] = arr[m + 1 + b];
16
17    a = 0;
18    b = 0;
19    c = l;
20    while (a < n1 && b < n2)
21    {
22        if (L[a] <= R[b])
23        {
24            arr[c] = L[a];
25            a++;
26        }
```

The bottom panel shows the 'Compilation results...' window with the following output:

```
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\user\Downloads\bubble sort.exe
- Output Size: 1,83314418792725 MiB
- Compilation Time: 3,81s
```

The status bar at the bottom indicates 'Line: 1 Col: 1 Sek: 0 Lines: 86 Length: 1485 Insert Done parsing in 0,032 seconds'.

Dimana Metode ada penggabungan biasanya digunakan pada untuk pengurutan berkas. Prinsip dari metode penggabungan ini sebagai berikut : mula-mula diberikan dua kumpulan data yang sudah dalam keadaan urut. Kedua kumpulan data tersebut harus dijadikan satu table sehingga dalam keadaan urut.

Kesimpulan yang kita bahasa dari kelompok kami

1. Mega sort yang tercepat dikarenakan sangat membutuhkan waktu yang sedikit ketika dalam melakukan sorting dan setiap computer juga memiliki spesifikasi yang berbeda-beda
2. bubble sort alasan merupakan metode yang sangat lambat dari beberapa bubble sort yang kita kerjakan ketika terjadi pengurutan bubble sort akan