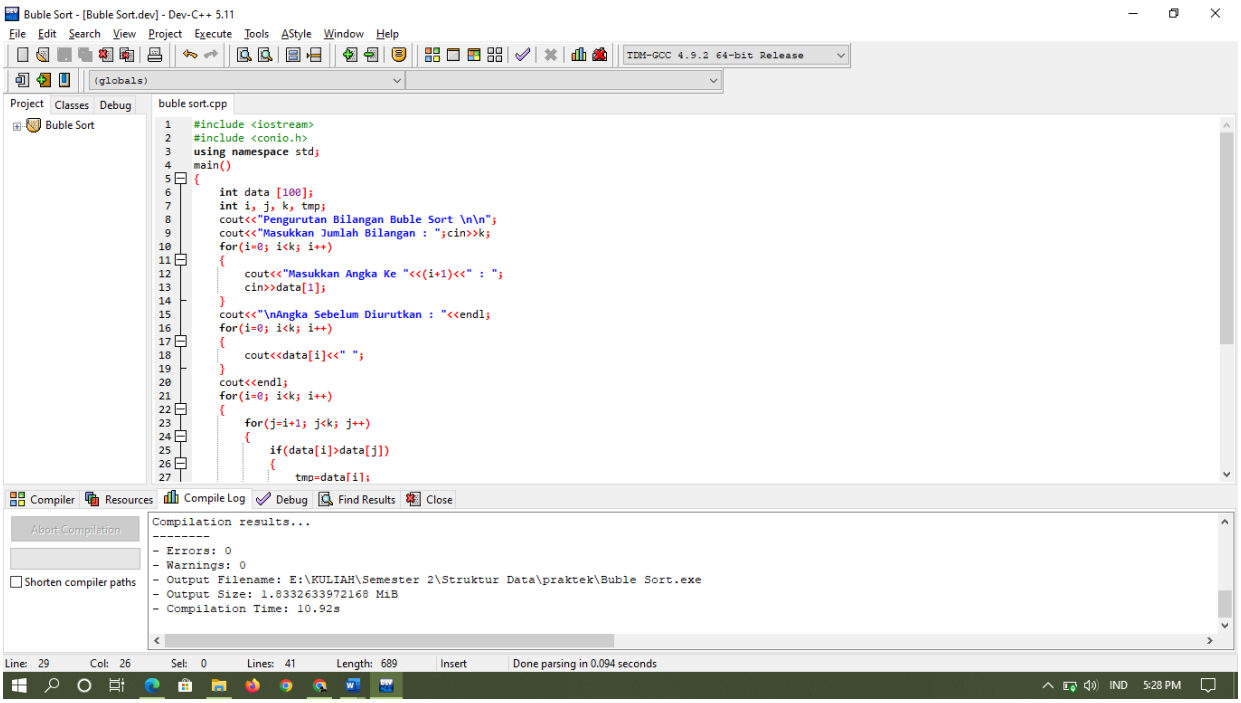
LAPORAN INDIVIDU
BUBLE SORT

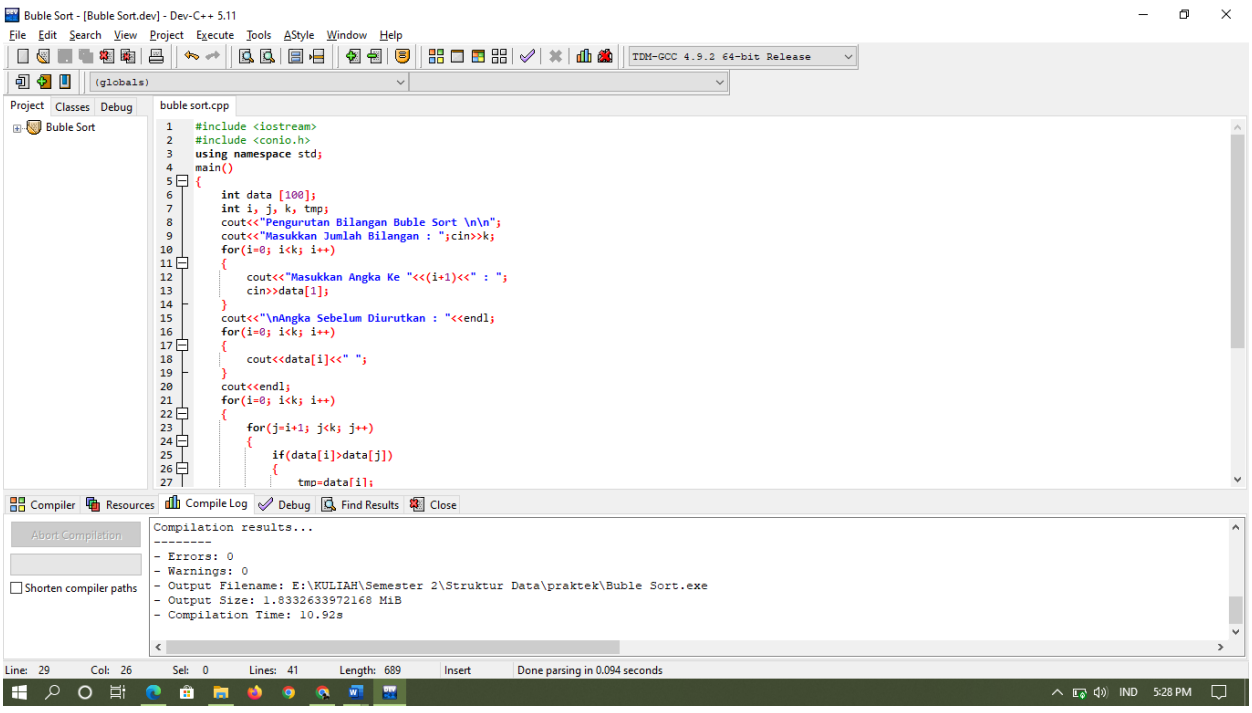
Muhammad Dhafa Jawadil Ubaid
21091397058
D4 Manajemen Informatika

Codingan Buble Sort

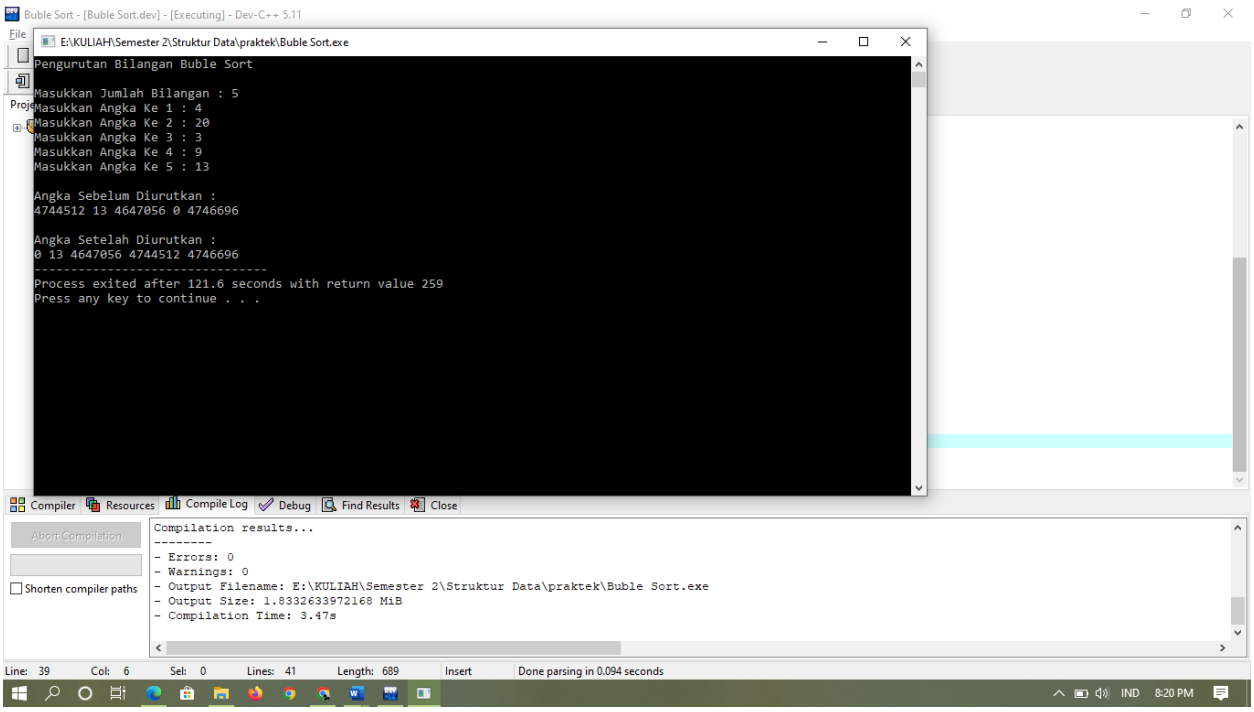
```
#include <iostream>
#include <conio.h>
using namespace std;
main()
{
	int data [100];
	int i, j, k, tmp;
	cout<<"Pengurutan Bilangan Buble Sort \n\n";
	cout<<"Masukkan Jumlah Bilangan : ";cin>>k;
	for(i=0; i<k; i++)
	{
		cout<<"Masukkan Angka Ke "<<(i+1)<<" : ";
		cin>>data[1];
	}
	cout<<"\nAngka Sebelum Diurutkan : "<<endl;
	for(i=0; i<k; i++)
	{
		cout<<data[i]<<" ";
	}
	cout<<endl;
	for(i=0; i<k; i++)
	{
		for(j=i+1; j<k; j++)
		{
			if(data[i]>data[j])
			{
				tmp=data[i];
				data[i]=data[j];
				data[j]=tmp;
			}
		}
	}
		cout<<"\nAngka Setelah Diurutkan : "<<endl;
	for(i=0; i<k; i++)
	{
		{
			cout<<data[i]<<" ";
		}
	}
	getch();
}
```


Buble Sort - [Buble Sort.dev] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
Buble Sort
buble sort.cpp
#include <iostream>
#include <conio.h>
using namespace std;
main()
{
int data [100];
int i, j, k, tmp;
cout<<"Pengurutan Bilangan Buble Sort \n\n";
cout<<"Masukkan Jumlah Bilangan : ";cin>>k;
for(i=0; i<k; i++)
{
cout<<"Masukkan Angka Ke "<<(i+1)<<" : ";
cin>>data[1];
}
cout<<"\nAngka Sebelum Diurutkan : "<<endl;
for(i=0; i<k; i++)
{
cout<<data[i]<<" ";
}
cout<<endl;
for(i=0; i<k; i++)
{
for(j=i+1; j<k; j++)
{
if(data[i]>data[j])
{
tmp=data[i];
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: E:\KULIAH\Semester 2\Struktur Data\praktek\Buble Sort.exe
- Output Size: 1.8332633972168 MiB
- Compilation Time: 10.92s
Line: 29 Col: 26 Sel: 0 Lines: 41 Length: 689 Insert Done parsing in 0.094 seconds
IND 5:28 PM


Buble Sort - [Buble Sort.dev] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
Buble Sort
buble sort.cpp
#include <iostream>
#include <conio.h>
using namespace std;
main()
{
int data [100];
int i, j, k, tmp;
cout<<"Pengurutan Bilangan Buble Sort \n\n";
cout<<"Masukkan Jumlah Bilangan : ";cin>>k;
for(i=0; i<k; i++)
{
cout<<"Masukkan Angka Ke "<<(i+1)<<" : ";
cin>>data[1];
}
cout<<"\nAngka Sebelum Diurutkan : "<<endl;
for(i=0; i<k; i++)
{
cout<<data[i]<<" ";
}
cout<<endl;
for(i=0; i<k; i++)
{
for(j=i+1; j<k; j++)
{
if(data[i]>data[j])
{
tmp=data[i];
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: E:\KULIAH\Semester 2\Struktur Data\praktek\Buble Sort.exe
- Output Size: 1.8332633972168 MiB
- Compilation Time: 10.92s
Line: 29 Col: 26 Sel: 0 Lines: 41 Length: 689 Insert Done parsing in 0.094 seconds
IND 5:28 PM



Penjelasan

Bubble sort adalah sebuah teknik pengurutan data dengan cara menukar dua data yang bersebelahan jika urutan dari data tersebut salah. Algorithma ini dapat mengurutkan data dari besar ke kecil (Ascending) dan kecil ke besar (Descending). Algoritma ini tidak cocok untuk set data dengan jumlah besar karena kompleksitas dari algorithma ini adalah O() di mana n adalah jumlah item.
Untuk belajar algoritma Bubble Sort ini kita hanya perlu memahami cara yang digunakan untuk mengurutkan data, sederhananya algoritma ini menggunakan perbandingan dalam operasi antar elemennya.
Dibawah ini merupakan gambaran dari algoritma Buble Sort dengan array “4,20,3,9,13”

Proses Pertama
(4,20,3,9,13) menjadi (4,20,3,9,13)
(4,20,3,9,13) menjadi (4,3,20,9,13)
(4,3,20,9,13) menjadi (4,3,9,20,13)
(4,3,9,20,13) menjadi (4,3,9,13,20)

Proses Kedua
(4,3,9,13,20) menjadi (3,4,9,13,20)
(3,4,9,13,20) menjadi (3,4,9,13,20)
(3,4,9,13,20) menjadi (3,4,9,13,20)
(3,4,9,13,20) menjadi (3,4,9,13,20)

MENGHITUNG BIG O KOMPLEKSITAS WAKTU

Kompleksitas waktu yang detil dari suatu algoritma. Biasanya yang kita butuhkan hanyalah hampiran dari kompleksitas waktu yang sebenarnya. Kompleksitas waktu ini dinamakan kompleksitas waktu asimptotik yang dinotasikan dengan “O” (baca : “O-besar”). Kompleksitas waktu asimptotik ini diperoleh dengan mengambil term terbesar dari suatu persamaan kompleksitas waktu. Sebagai contoh, dapat dilihat pada persamaan di bawah ini. T(n)=4n3+5n2+7n+3 (1) O(n3) (2) Dari persamaan (1) di atas diperoleh persamaan (2). Dapat dilihat bahwa nilai O adalah term terbesar dari T(n), tanpa faktor pengalinya. Berikut ini adalah daftar dari beberapa kelompok algoritma berdasarkan nilai O nya. Kelompok Algoritma Nama O(1) O(log n) O(n) O(n log n) O(n2) O(n3) O(2n) O(n!) konstan logaritmik lanjar n log n kuadratik kubik eksponensial faktorial Tabel 1. Pengelompokan Algoritma Berdasarkan Notasi OBesar Kompleksitas algoritma tersebut memiliki suatu spektrum, yang menunjukkan tingkat kompleksitas suatu algoritma, dengan urutan sebagai berikut.
O(1)<O(n)<…<O(n!)

Kelebihan :
a) proses perhitungan dengan metode yang paling sederhana,
b) mudah dipahami, dan
c) tahapan dalam pengurutan data sangat sederhana

Kekurangan :
a) proses perhitungan menggunakan metode pengurutan yang paling tidak efisien meskipun sederhana,
b) proses perhitungan akan lambat atau lama apabila data dengan jumlah besar. Jadi untuk proses pengurutan data secara tungal
c) jumlah pengulangan akan tetap sama sampai data yang terakhir, messkipun sebagian data ada telah diurut