TUGAS PRAKTIKUM STRUKTUR DATA

JOBSHEET 10

Dosen Pengampu:

Randi Proska Sandra, S.Pd, M.Sc

Kode Kelas:

202323430158

Disusun Oleh:

Wahyu Abdil Afif

23343085

PROGRAM STUDI INFORMATIKA (NK)

FAKULTAS TEKNIK

UNIVERSITAS NEGERI PADANG

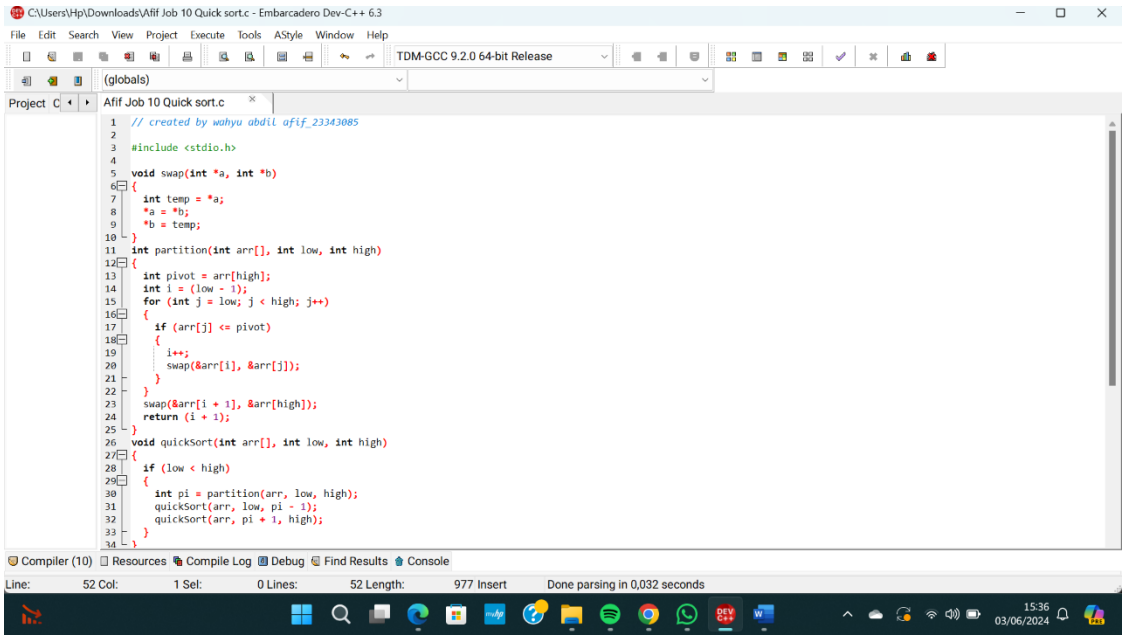
2024

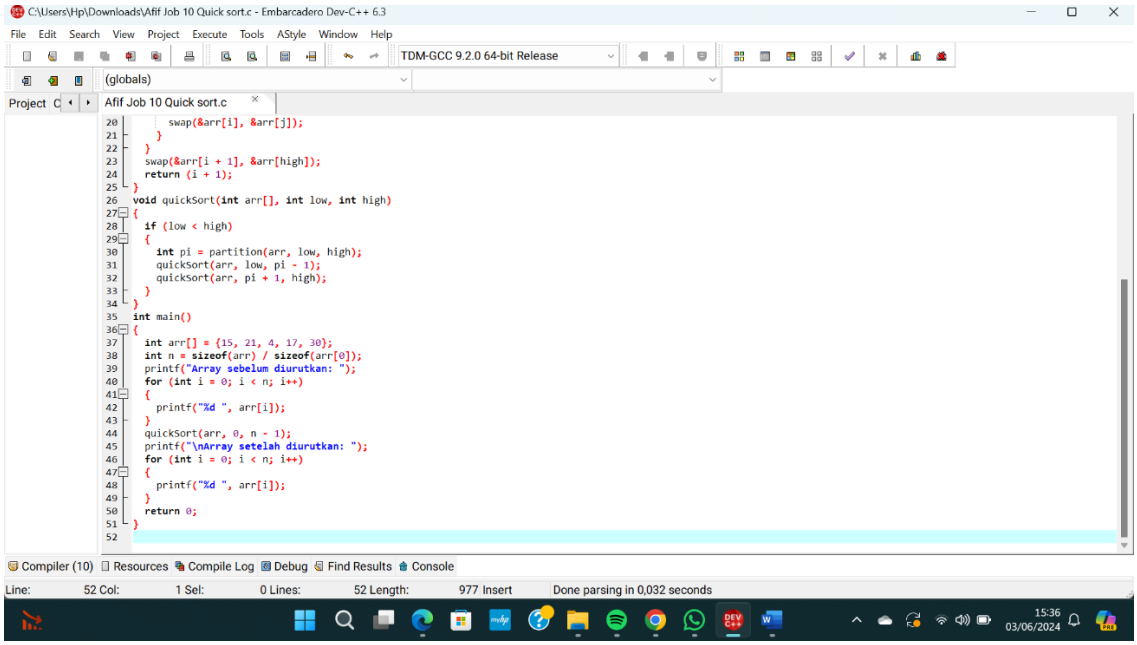
1. Quick sort

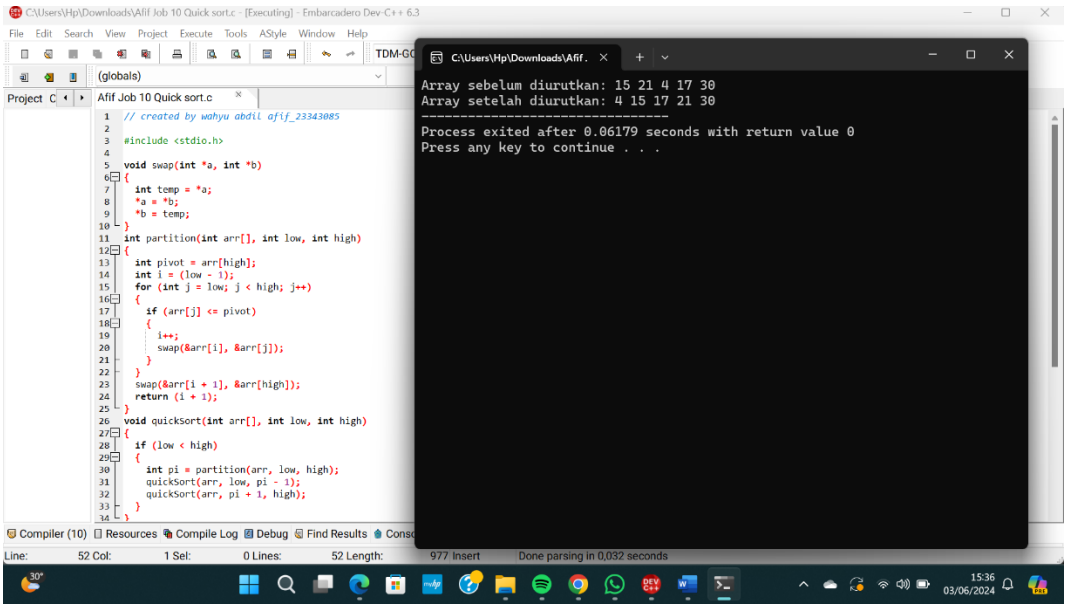
   Source code:

```
// created by wahyu abdil afif_23343085

#include <stdio.h>

void swap(int *a, int *b)
{
  int temp = *a;
  *a = *b;
  *b = temp;
}
int partition(int arr[], int low, int high)
{
  int pivot = arr[high];
  int i = (low - 1);
  for (int j = low; j < high; j++)
  {
    if (arr[j] <= pivot)
     {
      i++;
      swap(&arr[i], &arr[j]);
    }
  }
  swap(&arr[i + 1], &arr[high]);
  return (i + 1);
}
void quickSort(int arr[], int low, int high)
{
  if (low < high)
  {
    int pi = partition(arr, low, high);
    quickSort(arr, low, pi - 1);
    quickSort(arr, pi + 1, high);
  }
}
int main()
{
  int arr[] = {15, 21, 4, 17, 30};
  int n = sizeof(arr) / sizeof(arr[0]);
  printf("Array sebelum diurutkan: ");
  for (int i = 0; i < n; i++)
  {
    printf("%d ", arr[i]);
  }
  quickSort(arr, 0, n - 1);
  printf("\nArray setelah diurutkan: ");
```

```
  for (int i = 0; i < n; i++)
  {
    printf("%d ", arr[i]);
  }
  return 0;
}
```







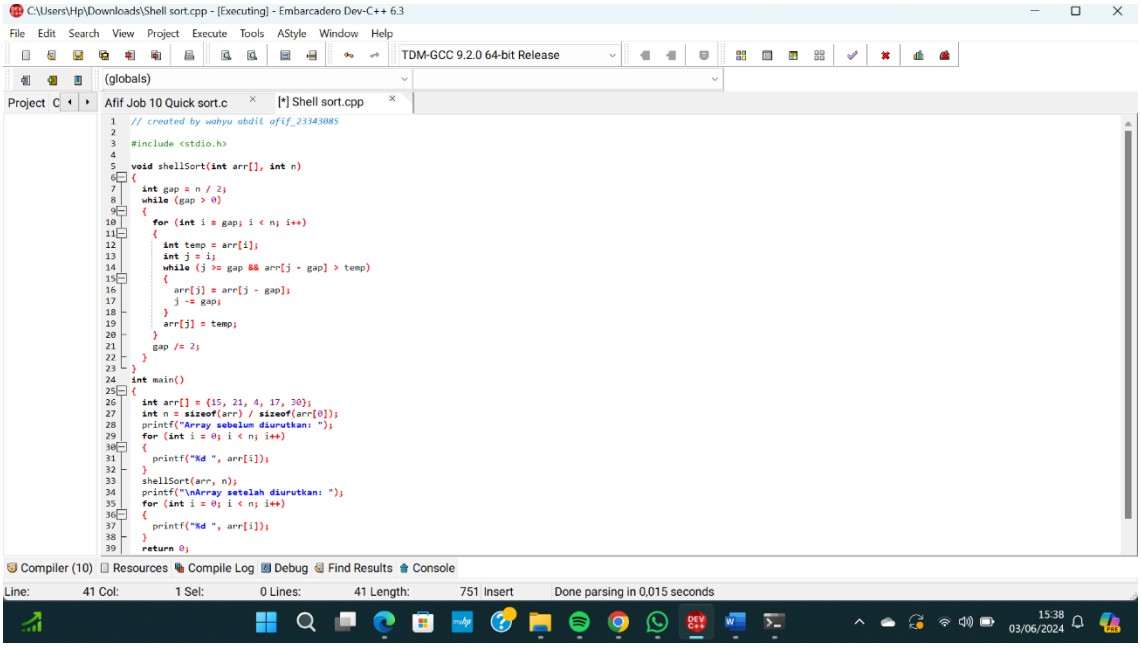
Penjelasan:

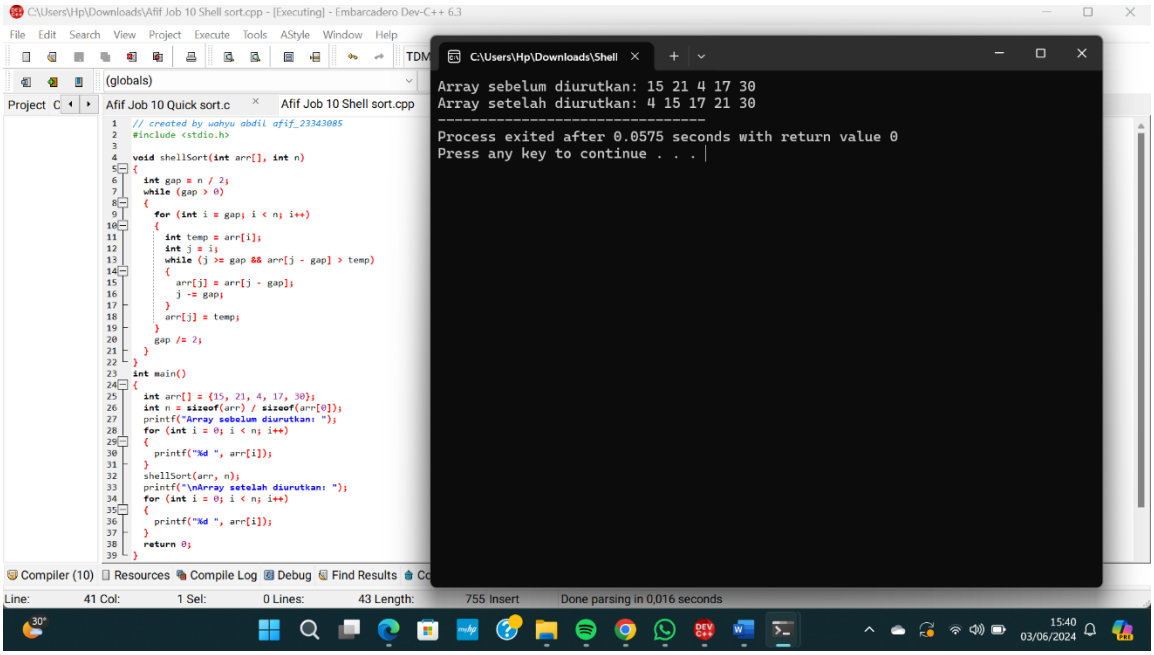
- Metode quick sort menggunakan pendekatan rekursif untuk membagi array menjadi dua sub-array berdasarkan suatu elemen pivot.
- Setiap elemen dalam array dibandingkan dengan pivot, dan elemen-elemen yang lebih kecil dari pivot ditempatkan di sebelah kiri pivot, sedangkan elemen-elemen yang lebih besar ditempatkan di sebelah kanan.
- Proses ini dilakukan secara rekursif untuk kedua sub-array hingga seluruh array diurutkan.
- Quicksort adalah pilihan yang baik untuk mengurutkan data berukuran besar karena efisiensinya. Namun, untuk data berukuran kecil atau data yang sudah terurut, algoritma lain seperti merge sort atau insertion sort mungkin lebih cocok. Penting untuk mempertimbangkan karakteristik data dan kebutuhan aplikasi saat memilih algoritma pengurutan.

2. Shell Sort

source code:

```
// created by wahyu abdil afif_23343085
#include <stdio.h>

void shellSort(int arr[], int n)
{
  int gap = n / 2;
  while (gap > 0)
  {
    for (int i = gap; i < n; i++)
     {
      int temp = arr[i];
      int j = i;
      while (j >= gap && arr[j - gap] > temp)
       {
        arr[j] = arr[j - gap];
        j -= gap;
      }
      arr[j] = temp;
    }
    gap /= 2;
  }
}
int main()
{
  int arr[] = {15, 21, 4, 17, 30};
  int n = sizeof(arr) / sizeof(arr[0]);
  printf("Array sebelum diurutkan: ");
  for (int i = 0; i < n; i++)
  {
    printf("%d ", arr[i]);
  }
  shellSort(arr, n);
  printf("\nArray setelah diurutkan: ");
  for (int i = 0; i < n; i++)
  {
    printf("%d ", arr[i]);
  }
  return 0;
}
```


C:\Users\Hp\Downloads\Shell sort.cpp - [Executing] - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 9.2.0 64-bit Release
(globals)
Project C
Afif Job 10 Quick sort.c
[*] Shell sort.cpp
// created by wahyu abdil afif_23343085
#include <stdio.h>
void shellSort(int arr[], int n)
{
int gap = n / 2;
while (gap > 0)
{
for (int i = gap; i < n; i++)
{
int temp = arr[i];
int j = i;
while (j >= gap && arr[j - gap] > temp)
{
arr[j] = arr[j - gap];
j -= gap;
}
arr[j] = temp;
}
gap /= 2;
}
}
int main()
{
int arr[] = {15, 21, 4, 17, 30};
int n = sizeof(arr) / sizeof(arr[0]);
printf("Array sebelum diurutkan: ");
for (int i = 0; i < n; i++)
{
printf("%d ", arr[i]);
}
shellSort(arr, n);
printf("\nArray setelah diurutkan: ");
for (int i = 0; i < n; i++)
{
printf("%d ", arr[i]);
}
return 0;
}
Compiler (10) Resources Compile Log Debug Find Results Console
Line: 41 Col: 1 Sel: 0 Lines: 41 Length: 751 Insert
Done parsing in 0,015 seconds
15:38
03/06/2024


C:\Users\Hp\Downloads\Afif Job 10 Shell sort.cpp - [Executing] - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project C
Afif Job 10 Quick sort.c
Afif Job 10 Shell sort.cpp
// created by wahyu abdil afif_23343085
#include <stdio.h>
void shellSort(int arr[], int n)
{
int gap = n / 2;
while (gap > 0)
{
for (int i = gap; i < n; i++)
{
int temp = arr[i];
int j = i;
while (j >= gap && arr[j - gap] > temp)
{
arr[j] = arr[j - gap];
j -= gap;
}
arr[j] = temp;
}
gap /= 2;
}
}
int main()
{
int arr[] = {15, 21, 4, 17, 30};
int n = sizeof(arr) / sizeof(arr[0]);
printf("Array sebelum diurutkan: ");
for (int i = 0; i < n; i++)
{
printf("%d ", arr[i]);
}
shellSort(arr, n);
printf("\nArray setelah diurutkan: ");
for (int i = 0; i < n; i++)
{
printf("%d ", arr[i]);
}
return 0;
}
C:\Users\Hp\Downloads\Shell
Array sebelum diurutkan: 15 21 4 17 30
Array setelah diurutkan: 4 15 17 21 30
Process exited after 0.0575 seconds with return value 0
Press any key to continue . . .
Compiler (10) Resources Compile Log Debug Find Results
Line: 41 Col: 1 Sel: 0 Lines: 43 Length: 755 Insert
Done parsing in 0,016 seconds
15:40
03/06/2024

Penjelasan:

- Metode shell sort merupakan variasi dari metode insertion sort.
- Ia menggunakan konsep "gap" untuk membagi array menjadi sub-grup yang lebih kecil.
- Setelah array dibagi, setiap sub-grup diurutkan secara terpisah menggunakan insertion sort.
- Kemudian, gap dikurangi secara bertahap hingga menjadi 1 dan array diurutkan sepenuhnya.
- Shell sort adalah pilihan yang baik untuk mengurutkan data yang berukuran sedang. Ini lebih efisien daripada insertion sort standar dan dapat bersaing dengan merge sort dan quicksort dalam kasus tertentu. Namun, pemilihan algoritma sorting yang tepat tergantung pada karakteristik data dan kebutuhan aplikasi.