TUGAS PERTEMUAN 11

STRUKTUR DATA

SORTING (QUICK SORT)

Disusun oleh:

Rama Pramudya Wibisana 2022320019

PROGRAM STUDI SISTEM INFORMASI

FAKULTAS INFORMATIKA

UNIVERSITAS BINA INSANI

BEKASI

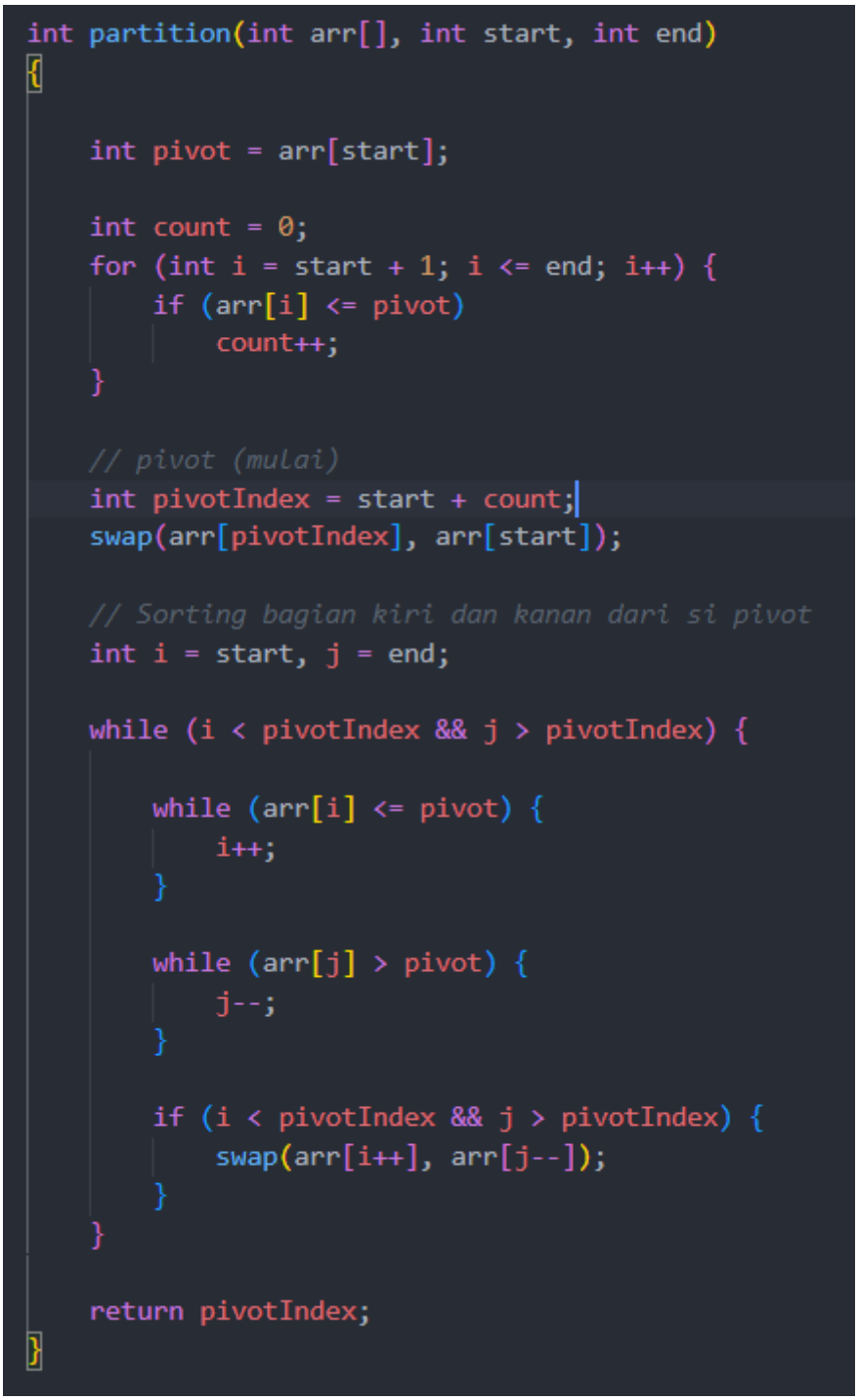
2022

A. Quick Sort Syntax

```cpp
// sorting tapi quicksort
#include <iostream>
using namespace std;

int partition(int arr[], int start, int end)
{

    int pivot = arr[start];

    int count = 0;
    for (int i = start + 1; i <= end; i++) {
        if (arr[i] <= pivot)
            count++;
    }

    // pivot (mulai)
    int pivotIndex = start + count;
    swap(arr[pivotIndex], arr[start]);

    // Sorting bagian kiri dan kanan dari si pivot
    int i = start, j = end;

    while (i < pivotIndex && j > pivotIndex) {

        while (arr[i] <= pivot) {
            i++;
        }

        while (arr[j] > pivot) {
            j--;
        }

        if (i < pivotIndex && j > pivotIndex) {
            swap(arr[i++], arr[j--]);
        }
    }

    return pivotIndex;
}
```

B. Penjelasan

Pada tugas kali ini, kita akan membahas sedikit mengenai cara untuk sorting sebuah array pada *c++.* Kali ini yang akan kita gunakan sebagai contoh yakni Quick Sorting. Mari kita langsung masuk ke pembahasan mengenai itu.



```
int partition(int arr[], int start, int end)
{

    int pivot = arr[start];

    int count = 0;
    for (int i = start + 1; i <= end; i++) {
        if (arr[i] <= pivot)
            count++;
    }

    // pivot (mulai)
    int pivotIndex = start + count;
    swap(arr[pivotIndex], arr[start]);

    // Sorting bagian kiri dan kanan dari si pivot
    int i = start, j = end;

    while (i < pivotIndex && j > pivotIndex) {

        while (arr[i] <= pivot) {
            i++;
        }

        while (arr[j] > pivot) {
            j--;
        }

        if (i < pivotIndex && j > pivotIndex) {
            swap(arr[i++], arr[j--]);
        }
    }

    return pivotIndex;
}
```

Kita membuat fungsi partition dengan tipe data integer. Fungsi ini digunakan untuk mempartisi element berdasarkan kondisi yang disebutkan dalam argument nya. Memiiki 3 parameter yaitu ***arr[],*** ***start*** & ***end*** yang masing masing memiliki tipe data integer.

Selanjutnya kita membuat variable pivot, yang menyimpan parameter arr[start]. Fungsi ini nantinya akan digunakan untuk membuat pivot awal. Lalu ada variable count dengan tipe data integer, mempunyai initial value 0; lakukan looping dengan membuat variable **I** dengan nilai awal start atau 0, kemudian + 1. Lalu dibuat kondisi jika I kurang sama dengan dari **end** maka I akan terus ditambah .

```
for (int i = start + 1; i <= end; i++) {
    if (arr[i] <= pivot)
        count++;
}
```

lalu, beri yang akan di loop adalah arr dengan parameter I dan jikai **arr[i]** kurang dari sama dengan pivot. Maka memanggil nilai awal dari variable count yang initial value nya = 0.

Setelah selesai dengan itu, setting titik awal dengan membuat variable pivotIndex = start + **count**;

Fungsi swap adalah fungsi yang digunakan untuk menukarkan dua buah nilai. Fungsi ini sangat dibutuhkan ketika kita hendak membuat aplikasi pengurutan data (sorting).

```
int pivotIndex = start + count;
swap(arr[pivotIndex], arr[start]);

// Sorting bagian kiri dan kanan dari si pivot
int i = start, j = end;

while (i < pivotIndex && j > pivotIndex) {

    while (arr[i] <= pivot) {
        i++;
    }

    while (arr[j] > pivot) {
        j--;
    }

    if (i < pivotIndex && j > pivotIndex) {
        swap(arr[i++], arr[j--]);
    }
}

return pivotIndex;
```

Lalu mulai sorting bagian kiri kanan dengan menyimpan nya pada variable I dan j, I untuk mulai (kiri) dan J untuk selesai (kanan). Buat percabangan dengan while, jika I kurang dari pivotIndex yang tadi isinya adalah start + count.gunakan operator logika && (dan) maka di bac ajika I kurang dari pivotIndex dan J lebih dari pivotIndex, seperti itu. Buat lagi percabangan dengan while, kali ini kita gunakan nested while atau percabangan rumit :D.

Apabila arr[i] <= pivot maka akan mengembalikan nilai increament, atau nilai akan ditambah. Namun jika arr[j] (end) sudah lebih besar dari pivot, maka akan di kurangi. Lalu akan kita gunakan lagi fungsi swap untuk menukar dua buah nilai tadi I dan j. gunakan fungsi if jika I < pivotIndex dan j lebih dari pivotIndex, maka ditukar nilai dari arr[i++] dan arr[j--].

```
void quickSort(int arr[], int start, int end)
{

    // base case
    if (start >= end)
        return;

    // partitioning the array
    int p = partition(arr, start, end);

    // Sorting the left part
    quickSort(arr, start, p - 1);

    // Sorting the right part
    quickSort(arr, p + 1, end);
}
```

Lalu kita mulai masuk ke menu utamanya, kita buat fungsi quicksort dengan tipe data void yang mana tidak perlu mengembalikan nilai / value pada akhir kode. Mempunyai parameter yang sama seperti diatas yaitu arr, start dan end dengan masing masing memiliki tipe data integer. Untuk membuat standart nya, kita tentukan dili jika start lebih dari sama dengan end, maka tidak akan mengembalikan nilai apapun. Kemudian kita buat variable p untuk menyimpan fungsi partition yang tadi telah kita buat, jangan lupa panggil parameternya. Setelah nya kita buat perintah sorting bagian kiri kanan dengan p akan dikurang 1 untuk bagian kiri, dan di tambah 1 untuk bagian kanan.

```
int main()
{

    int arr[] = { 1, 5, 8, 9, 6, 7, 3, 4, 2, 0};
    int n = 10;

    quickSort(arr, 0, n - 1);
    cout<<"Bentuk Array setelah dilakukan quicksort: "<<endl;
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }

    return 0;
}
```

Terakhir.. kita buat fungsi main untuk menampilkan apa yang sudah kita buat diatas. kita akan menentukan nilai yang akan di urutkan, maka buat variable arr[] dengan nilai berbentuk integer yang acak. Lalu buat n sebagai pivot, kita akan set dia 1 angka setelah angka tertinggi dari nilai yang sudah di tentukan di variable arr[]. Untuk menjalankan nya panggil fungsi quicksort, masukan parameter yang diminta seperti variable nya, nilai awal dan p atau pivot yang dimana disini kita buat dengan variable n.

Maka lakukan perulangan sederhana, set nilai I dengan 0, dan jika I kurang dari n, maka I akan ditambah. Lalu kembalikan dengan variable arr[], namun kali ini panggil I untuk mengrutukan element nya yang sudah kita loop. Maka quicksort akan seperti ini.

Before :

```
int arr[] = { 1, 5, 8, 9, 6, 7, 3, 4, 2, 0};
```

After:

```
Bentuk Array setelah dilakukan quicksort:
0 1 2 3 4 5 6 7 8 9
```