LAPORAN PRAKTIKUM STRUKTUR DATA DAN ALGORITMA

MODUL 8

ALGORITMA SEARCHING

DISUSUN OLEH:

PRIESTY AMEILIANA MAULIDAH

2311102175

S1 IF-11-E

DOSEN:

Muhammad Afrizal Amrustian, S. Kom.

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO PURWOKERTO

2024

A.DASAR TEORI

Pencarian, atau searching, adalah proses menemukan nilai tertentu dalam kumpulan data. Hasil dari pencarian dapat berupa data yang ditemukan, data yang ditemukan lebih dari satu, atau ketidakadaan data. Pencarian dilakukan dengan memeriksa setiap elemen dalam array atau matriks menggunakan perulangan. Ada dua metode pencarian dalam algoritma searching, yaitu Sequential Search dan Binary Search.

Sequential Search adalah algoritma pencarian yang digunakan untuk data yang acak atau belum terurut. Prosesnya dilakukan dengan membandingkan setiap elemen satu per satu secara berurutan, dimulai dari indeks pertama hingga indeks terakhir. Jika data ditemukan, pencarian berhenti. Jika tidak, pencarian berlanjut hingga akhir array dan data tidak ditemukan.

Binary Search merupakan algoritma pencarian untuk data yang terurut. Pada metode ini, data dibagi menjadi dua bagian secara berulang, dan data yang dicari dibandingkan dengan data di tengah. Jika data lebih besar, maka pencarian dilakukan di bagian kanan. Jika data lebih kecil, pencarian dilakukan di bagian kiri. Proses ini terus dilakukan hingga data ditemukan atau tidak ditemukan.

Kedua algoritma pencarian ini dapat diimplementasikan dengan menggunakan langkah-langkah tertentu, seperti pengecekan elemen secara berurutan dalam Sequential Search dan pembagian data dalam Binary Search.

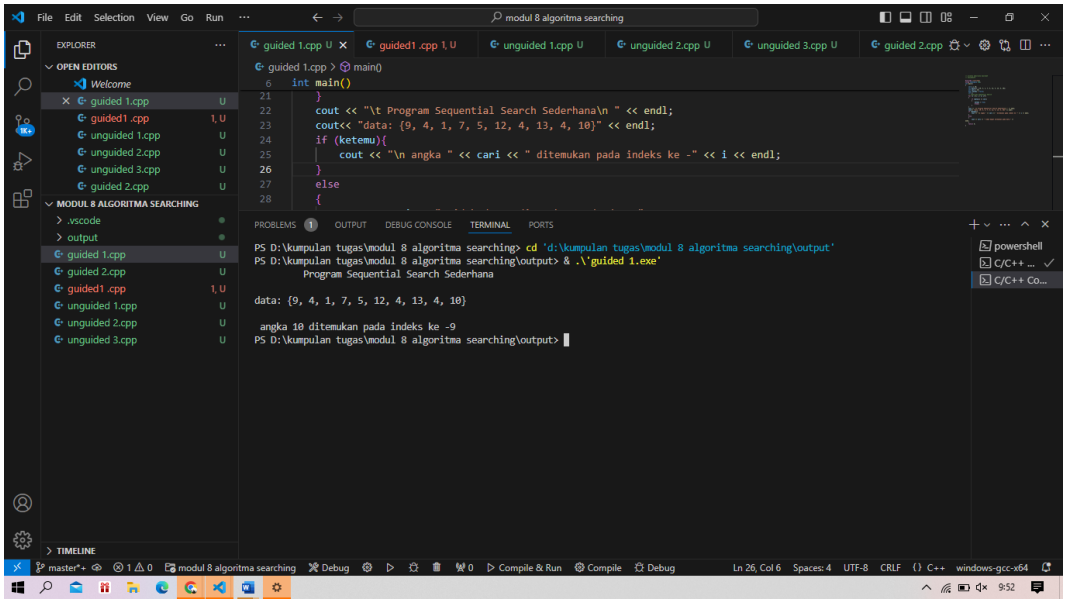
B.Guided

Guided 1

```
// priesty ameiliana maulidah
// 2311102175
#include <iostream>
using namespace std;
int main()
{
   int n = 10;
   int data[n] = {9, 4, 1, 7, 5, 12, 4, 13, 4, 10};
   int cari = 10;
   bool ketemu = false;
   int i;
   // algoritma Sequential Search
   for (i = 0; i < n; i++)
   {
      if (data[i] == cari)
      {
         ketemu = true;
         break;
      }
   }
```

```
cout << "\t Program Sequential Search Sederhana\n " << endl;
   cout<< "data: {9, 4, 1, 7, 5, 12, 4, 13, 4, 10}" << endl;
   if (ketemu){
      cout << "\n angka " << cari << " ditemukan pada indeks ke -" << i << endl;
   }
   else
   {
      cout << cari << " tidak dapat ditemukan pada data." <<
endl;
   }
   return 0;
}
```

Screenshots output



Deskripsi :

1. Deklarasi Variabel:

- n: Menampung jumlah elemen dalam array.
- data: Array yang menyimpan data yang akan dicari.
- cari: Nilai yang ingin dicari dalam array.
- ketemu: Boolean yang menandakan apakah nilai yang dicari telah ditemukan.
- i: Variabel iterator untuk loop.

2. Input Data:

- Menginisialisasi n dengan nilai 10.

- Menginisialisasi array data dengan 10 elemen berisi nilai-nilai integer.
- Menentukan nilai cari yang ingin dicari.

3. Algoritma Pencarian Sekuensial:

- Looping melalui array data menggunakan iterator i.
- Pada setiap iterasi, membandingkan nilai elemen array data[i] dengan nilai cari.
- Jika ditemukan elemen yang sama dengan cari, set ketemu menjadi true dan break dari loop.

4. Output Hasil:

- Menampilkan pesan header program.
- Menampilkan isi array data.
- Jika ketemu true, menampilkan pesan bahwa nilai cari ditemukan pada indeks ke-i.
- Jika ketemu false, menampilkan pesan bahwa nilai cari tidak ditemukan.

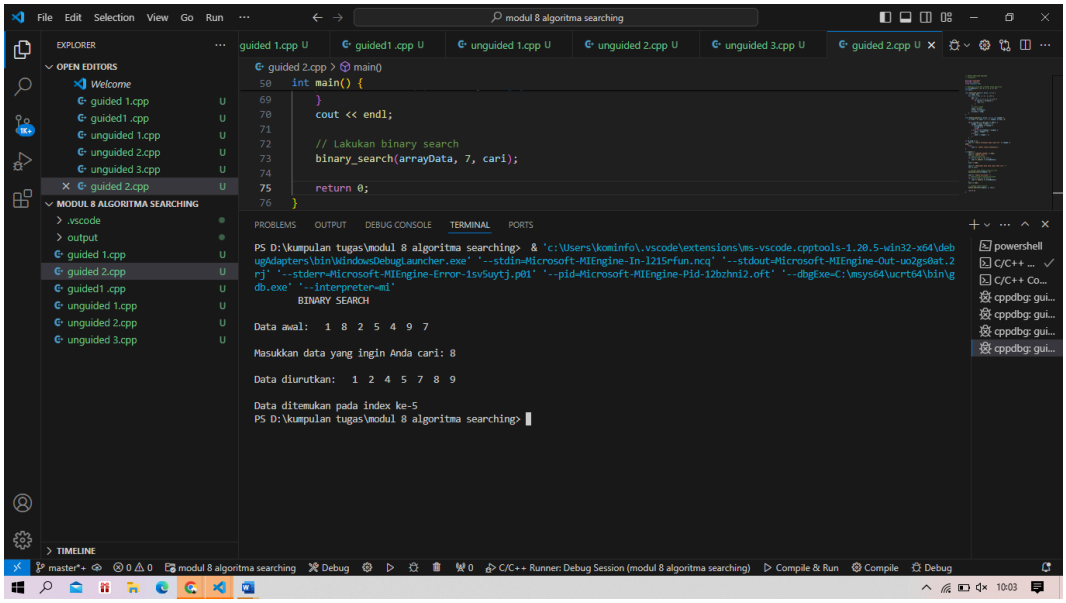
Guided 2

```
// PRIESY AMEILIANA MAULIDAH
// 2311102175

#include <iostream>
#include <iomanip>
using namespace std;

// Deklarasi array dan variabel untuk pencarian
int arrayData[7] = {1, 8, 2, 5, 4, 9, 7};
int cari;

void selection_sort(int arr[], int n) {
    int temp, min;
    for (int i = 0; i < n - 1; i++) {
        min = i;
        for (int j = i + 1; j < n; j++) {
            if (arr[j] < arr[min]) {
                min = j;
            }
        }
        // Tukar elemen
        temp = arr[i];
        arr[i] = arr[min];
        arr[min] = temp;
    }
}
```

```
void binary_search(int arr[], int n, int target) {
   int awal = 0, akhir = n - 1, tengah, b_flag = 0;

   while (b_flag == 0 && awal <= akhir) {
      tengah = (awal + akhir) / 2;
      if (arr[tengah] == target) {
         b_flag = 1;
         break;
      } else if (arr[tengah] < target) {
         awal = tengah + 1;
      } else {
         akhir = tengah - 1;
      }
   }

if (b_flag == 1)
   cout << "\nData ditemukan pada index ke-" << tengah <<
endl;
   else
      cout << "\nData tidak ditemukan\n";
}
```

```
int main() {
    cout << "\tBINARY SEARCH" << endl;
    cout << "\nData awal: ";
    // Tampilkan data awal
    for (int x = 0; x < 7; x++) {
        cout << setw(3) << arrayData[x];
    }
    cout << endl;

    cout << "\nMasukkan data yang ingin Anda cari: ";
    cin >> cari;

    // Urutkan data dengan selection sort
    selection_sort(arrayData, 7);

    cout << "\nData diurutkan: ";
    // Tampilkan data setelah diurutkan
    for (int x = 0; x < 7; x++) {
        cout << setw(3) << arrayData[x];
    }
    cout << endl;

    // Lakukan binary search
    binary_search(arrayData, 7, cari);

    return 0;
}
```

Screenshots output:



Deskripsi :

1. Deklarasi Array dan Variabel:

- arrayData: Array yang menyimpan data yang akan dicari, berisi 7 elemen integer.
- cari: Nilai yang ingin dicari dalam array.

2. Fungsi selection_sort:

- Fungsi ini mengimplementasikan algoritma **Selection Sort** untuk mengurutkan array arrayData secara ascending.
- Algoritma ini bekerja dengan cara mencari nilai minimum dalam array pada setiap iterasi dan menukarnya dengan elemen pertama pada iterasi tersebut.

- Fungsi ini menerima 2 parameter:
    - arr: Array yang akan diurutkan.
    - n: Jumlah elemen dalam array.

3. Fungsi binary_search:

- Fungsi ini mengimplementasikan algoritma **Binary Search** untuk mencari nilai cari dalam array arrayData.
- Algoritma ini bekerja dengan cara membagi array menjadi dua bagian secara berulang, dan membandingkan nilai target dengan nilai tengah array.
- Jika nilai target sama dengan nilai tengah, maka nilai target telah ditemukan.
- Jika nilai target lebih kecil dari nilai tengah, maka pencarian dilanjutkan pada bagian kiri array.
- Jika nilai target lebih besar dari nilai tengah, maka pencarian dilanjutkan pada bagian kanan array.
- Fungsi ini menerima 3 parameter:
    - arr: Array yang sudah diurutkan.
    - n: Jumlah elemen dalam array.
    - target: Nilai yang ingin dicari.

4. Fungsi main:

- Fungsi main adalah fungsi utama program yang mengontrol alur program.
- Berikut langkah-langkah dalam fungsi main:
    1. Menampilkan judul program "BINARY SEARCH".
    2. Menampilkan data awal dalam array arrayData.
    3. Meminta pengguna untuk memasukkan nilai yang ingin dicari (cari).
    4. Mengurutkan array arrayData menggunakan fungsi selection_sort.
    5. Menampilkan data setelah diurutkan.
    6. Melakukan pencarian nilai cari dalam array yang telah diurutkan menggunakan fungsi binary_search.
    7. Menampilkan hasil pencarian (nilai ditemukan atau tidak ditemukan).

c. unguided/tugas

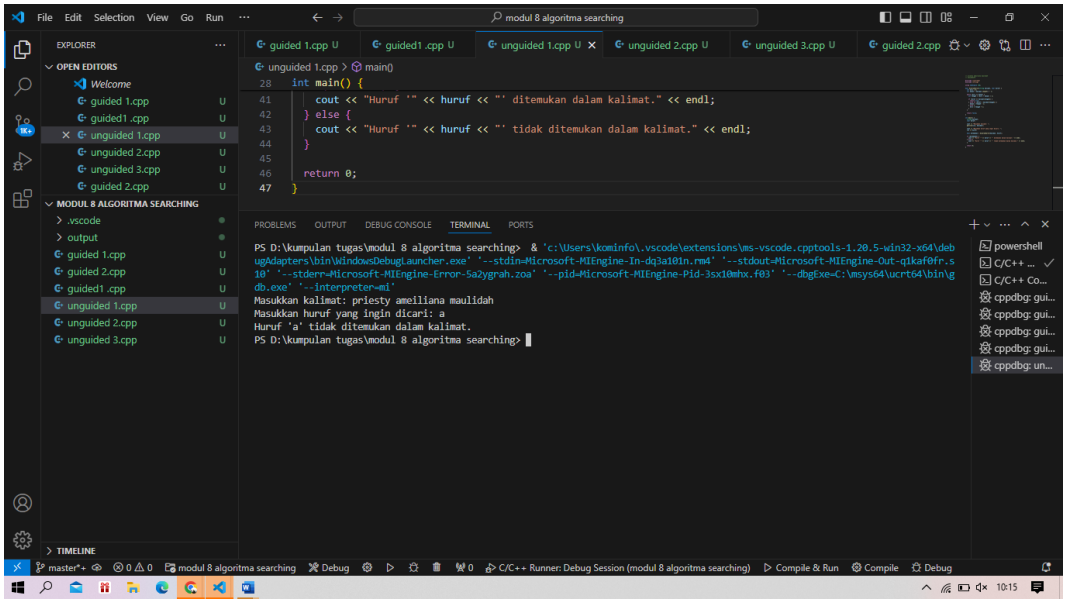
unguided 1

```
// priesty ameiliana maulidah
// 2311102175

#include <iostream>
#include <string>

using namespace std;

bool binarySearch(string kalimat, char huruf) {
  int kiri = 0;
  int kanan = kalimat.length() - 1;

  while (kiri <= kanan) {
    int tengah = (kiri + kanan) / 2;
```

```
if (huruf == kalimat[tengah]) {
      return true;
    } else if (huruf < kalimat[tengah]) {
      kanan = tengah - 1;
    } else {
      kiri = tengah + 1;
    }
  }

  return false;
}

int main() {
  string kalimat;
  char huruf;

  cout << "Masukkan kalimat: ";
  getline(cin, kalimat);

  cout << "Masukkan huruf yang ingin dicari: ";
  cin >> huruf;

  bool ditemukan = binarySearch(kalimat, huruf);

  if (ditemukan) {
```

```
cout << "Huruf '" << huruf << "' ditemukan dalam kalimat." << endl;
  } else {
    cout << "Huruf '" << huruf << "' tidak ditemukan dalam kalimat." << endl;
  }

  return 0;
}
```

Screenshots output:



Deskripsi :

1. Deklarasi Fungsi:

- binarySearch(string kalimat, char huruf):
    - Fungsi ini mengimplementasikan algoritma **Binary Search** untuk mencari huruf huruf dalam kalimat kalimat.
    - Fungsi ini mengembalikan nilai boolean true jika huruf ditemukan, dan false jika tidak ditemukan.
    - Fungsi ini menerima 2 parameter:
        - kalimat: String yang akan dicari.
        - huruf: Huruf yang ingin dicari.

2. Fungsi main:

- Fungsi main adalah fungsi utama program yang mengontrol alur program.
- Berikut langkah-langkah dalam fungsi main:
    1. Meminta pengguna untuk memasukkan kalimat.
    2. Meminta pengguna untuk memasukkan huruf yang ingin dicari.
    3. Memanggil fungsi binarySearch untuk mencari huruf dalam kalimat.
    4. Menampilkan hasil pencarian (huruf ditemukan atau tidak ditemukan).

3. Algoritma Pencarian:

- Algoritma **Binary Search** bekerja dengan cara membagi kalimat menjadi dua bagian secara berulang, dan membandingkan huruf yang dicari dengan huruf pada tengah kalimat.
- Jika huruf yang dicari sama dengan huruf pada tengah kalimat, maka huruf tersebut telah ditemukan.
- Jika huruf yang dicari lebih kecil dari huruf pada tengah kalimat, maka pencarian dilanjutkan pada bagian kiri kalimat.

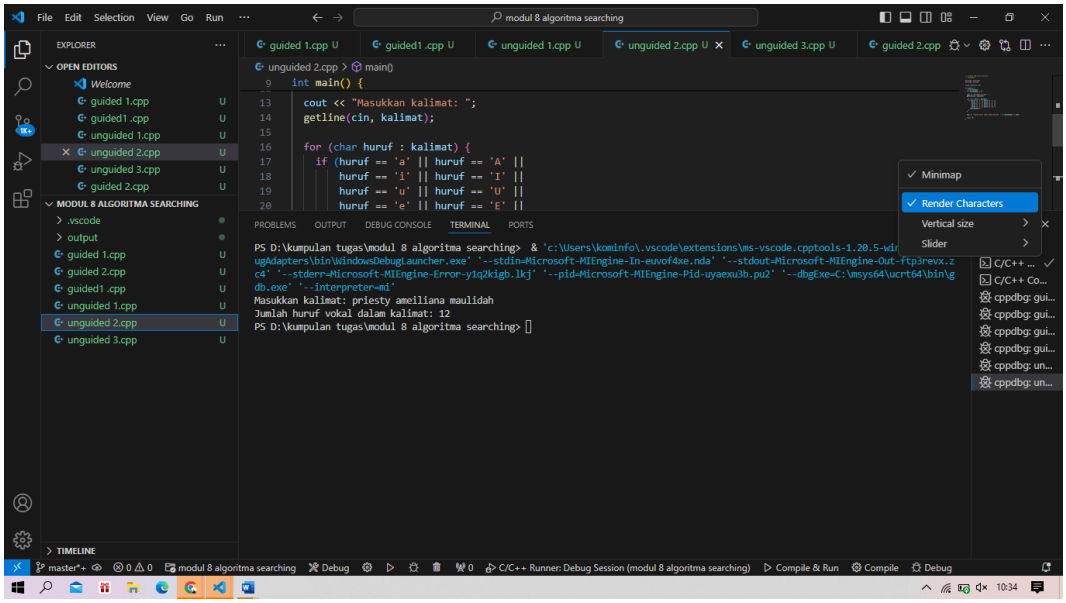
- Jika huruf yang dicari lebih besar dari huruf pada tengah kalimat, maka pencarian dilanjutkan pada bagian kanan kalimat.

Unguided 2

```
// priesty ameiliana maulidah
// 2311102175

#include <iostream>
#include <string>

using namespace std;

int main() {
  string kalimat;
  int jumlahVokal = 0;

  cout << "Masukkan kalimat: ";
  getline(cin, kalimat);

for (char huruf : kalimat) {
    if (huruf == 'a' || huruf == 'A' ||
        huruf == 'i' || huruf == 'I' ||
        huruf == 'u' || huruf == 'U' ||
        huruf == 'e' || huruf == 'E' ||
        huruf == 'o' || huruf == 'O') {
      jumlahVokal++;
    }
  }
```

```
cout << "Jumlah huruf vokal dalam kalimat: " << jumlahVokal << endl;


  return 0;

}
```

Screenshots output:



Deskripsi :

1. Fungsi main:

- Fungsi main adalah fungsi utama program yang mengontrol alur program.
- Berikut langkah-langkah dalam fungsi main:
    1. Meminta pengguna untuk memasukkan kalimat.
    2. Menginisialisasi variabel jumlahVokal untuk menyimpan jumlah huruf vokal.
    3. Melakukan iterasi melalui setiap huruf dalam kalimat menggunakan loop for each.

4. Di dalam loop, periksa apakah huruf saat ini adalah salah satu huruf vokal (a, A, i, I, u, U, e, E, o, O).
    - Jika ya, tambahkan 1 ke variabel jumlahVokal.
5. Setelah loop selesai, tampilkan jumlah huruf vokal dalam kalimat.

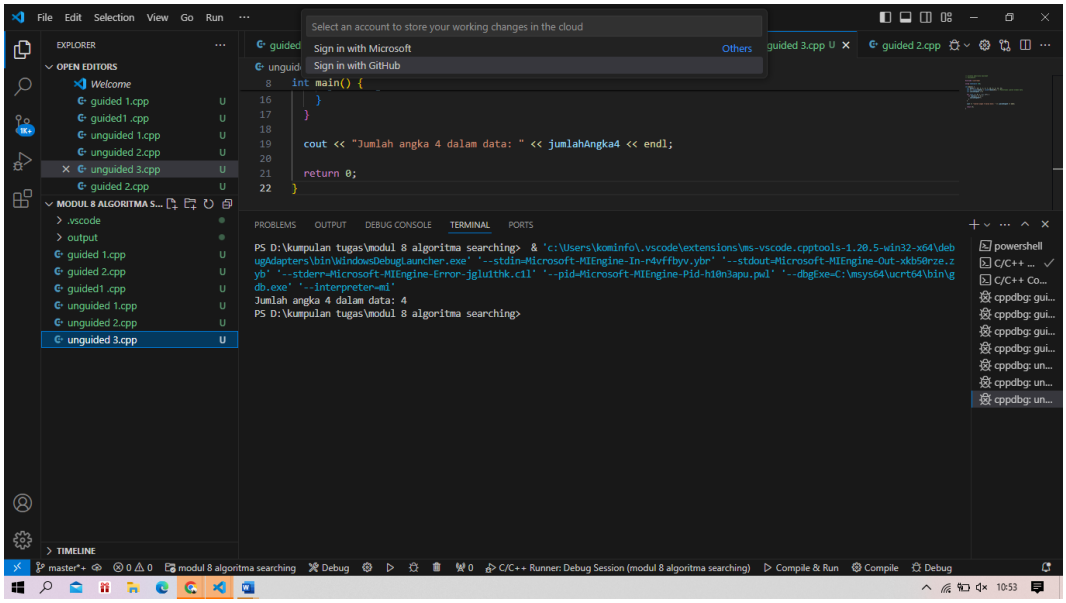
2. Perhitungan Vokal:

- Kode menggunakan loop for each untuk iterasi melalui setiap karakter dalam kalimat.
- Di dalam loop, kondisi if digunakan untuk memeriksa apakah karakter saat ini adalah salah satu huruf vokal.
    - Huruf vokal yang diperiksa adalah 'a', 'A', 'i', 'I', 'u', 'U', 'e', 'E', 'o', dan 'O'.
    - Jika karakter saat ini adalah huruf vokal, variabel jumlahVokal ditambah 1.

Unguided 3

```
// priesty ameiliana maulidah
// 2311102175

#include <iostream>

using namespace std;

int main() {
  int data[] = {9, 4, 1, 4, 7, 10, 5, 4, 12, 4};
  int n = sizeof(data) / sizeof(data[0]); // Menentukan jumlah elemen data
  int jumlahAngka4 = 0;

  for (int i = 0; i < n; i++) {
    if (data[i] == 4) {
      jumlahAngka4++;
    }
  }

  cout << "Jumlah angka 4 dalam data: " << jumlahAngka4 << endl;

  return 0;
}
```

Screenshots output:



Deskripsi :

1. Deklarasi Array dan Variabel:

- data: Array yang menyimpan data integer, berisi nilai-nilai {9, 4, 1, 4, 7, 10, 5, 4, 12, 4}.
- n: Variabel yang menyimpan jumlah elemen dalam array data.
- jumlahAngka4: Variabel yang menyimpan jumlah kemunculan angka 4.

2. Menentukan Jumlah Elemen Array:

- n = sizeof(data) / sizeof(data[0]): Ekspresi ini digunakan untuk menghitung jumlah elemen dalam array data.

    - sizeof(data): Mengembalikan ukuran keseluruhan array data dalam bytes.
    - sizeof(data[0]): Mengembalikan ukuran elemen tunggal dalam array data (dalam hal ini, ukuran integer).
    - Hasil pembagian ini memberikan jumlah elemen dalam array.

3. Looping dan Penghitungan:

- Loop for digunakan untuk iterasi melalui setiap elemen dalam array data.
- Di dalam loop, kondisi if digunakan untuk memeriksa apakah nilai elemen saat ini (data[i]) sama dengan 4.
    - Jika ya, variabel jumlahAngka4 ditambah 1.

4. Output:

- Setelah iterasi selesai, nilai jumlahAngka4 yang berisi jumlah kemunculan angka 4 ditampilkan ke konsol.

E. Referensi

https://naratif.sttbandung.ac.id/index.php/naratif/article/view/21