

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN**

**MODUL XI
“Dasar Pemrograman XI”**



**Disusun oleh:
Avriel Nur Adi Pratama
2311103158**

**Dosen Pengampu:
Toni Anwar S.Kom, M.Msi**

**PROGRAM STUDI SISTEM INFORMASI
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
PURWOKERTO
2023**

BAB I

TUJUAN PRAKTIKUM

1. Mahasiswa mampu menjelaskan konsep pencarian
2. Mahasiswa mampu dalam menjelaskan algoritme pencarian sequential
3. Mahasiswa mampu dalam penerapan algoritme pencarian sequential

BAB II

DASAR TEORI

Pencarian (searching) merupakan proses yang fundamental dalam pengolahan data. Proses pencarian adalah menemukan nilai (data) tertentu di dalam sekumpulan data yang bertipe sama (baik bertipe dasar atau bertipe bentukan). Sebagai contoh, untuk mengubah (update) data tertentu, langkah pertama yang harus dilakukan adalah mencari keberadaan data tersebut didalam kumpulannya. Jika data yang dicari ditemukan, maka data tersebut dapat diubah nilainya dengan data yang baru. Aktivitas awal yang sama juga dilakukan pada proses penambahan (insert) data baru. Proses penambahan data dimulai dengan mencari apakah data yang akan ditambahkan sudah terdapat di dalam kumpulan. Jika sudah ada dan mengasumsikan tidak boleh ada duplikasi data maka data tersebut tidak perlu ditambahkan, tetapi jika belum ada, maka tambahkan.

Suatu teknik pencarian data dalam array yang akan menelusuri semua elemenelemen array dari awal sampai akhir, dimana data-data tidak perlu diurutkan terlebih dahulu. Kemungkinan terbaik (best case) adalah jika data yang dicari terletak di indeks array terdepan (elemen array pertama) sehingga waktu yang dibutuhkan untuk pencarian data sangat sebentar (minimal). Kemungkinan terburuk (worst case) adalah jika data yang dicari terletak di indeks array terakhir (elemen array terakhir) sehingga waktu yang dibutuhkan untuk pencarian data sangat lama (maksimal).

Secara umum, algoritme pencarian beruntun berjalan lambat. Waktu pencarian sebanding dengan jumlah elemen larik. Misalkan larik berukuran 11 elemen. Maka, pada kasus di mana x tidak terdapat di dalam larik atau x ditemukan pada elemen yang terakhir, kita harus melakukan perbandingan dengan seluruh elemen larik, yang berarti jumlah perbandingan yang terjadi sebanyak n kali. Kita katakan bahwa waktu pencarian dengan algoritme pencarian beruntun sebanding dengan n . Bayangkan bila larik berukuran 100.000 buah elemen, maka kita harus melakukan perbandingan sebanyak 100.000 buah elemen. Andaikan satu operasi perbandingan elemen larik membutuhkan waktu 0.01 detik, maka untuk 100.000 buah perbandingan diperlukan waktu sebesar 1000 detik atau 16,7 menit.

BAB III

GUIDED

Tuliskan source code, screenshot dan juga deskripsi program hasil dari latihan yang **dipraktikan bersama** selama praktikum. (yang dikerjakan bersama-sama/bareng-bareng)


Contoh:

1. Guided 1

Source code

```
#include <iostream>
using namespace std;
int main(){
    const int panjangArray = 6;
    int data[panjangArray] = {13,16,14,21,76,15};
    int cari, penghitung = 0;
    bool ketemu=0;
    cout<<"Masukkan data yang ingin dicari = ";
    cin>>cari;
    for(int i=0; i<panjangArray; i++){
        penghitung++;
        if(data[i] == cari) {
            ketemu = 1;
            break;
        }
    }
    if(ketemu==1){
        cout<<"Data ada!" <<endl;
        cout<<"Proses perbandingan terjadi sebanyak : "<<penghitung<<endl;
    }else{
        cout<<"Data tidak ada!"<<endl;
    }
    return 0;
}
```

Screenshoot program



```
PS C:\Users\Said\Documents\Beasiswa\modul11> cd "c:\Users\Said\Documents\Beasiswa\modul11" ; if ($?) { g++ guided1.cpp -o guided1 } ; if ($?) { .\guided1 }
Masukkan data yang ingin dicari = 13
Data ada!
Proses perbandingan terjadi sebanyak : 1
```

Deskripsi program

- Panjang aarray 6 dengan tipe data integer. Berisi data 12,16,14,21,76,15
- Menjalankan for dari indeks 0 hingga indeks panjangarray
- Disetiap perulangan variabel akan bertambah 1
- Jika data pada i sama dengan yang dicara maka variabel ketemu akan bernilai 1 dan akan berhenti. Jika perulangan selese dan variabel masih 0, maka data yang dicari tidak ada

2. Guided 2

Source Code

```
#include <iostream>
using namespace std;
int main()
{
    const int panjangArray = 6;
    string data[panjangArray] = {"Avriel", "Bima", "Yanu", "Arsyad", "Habib",
    "Zalfa"};
    string cari;
    int penghitung = 0;
    bool ketemu = 0;
    cout << "Masukkan data yang ingin dicari = ";
    cin >> cari;
    for (int i = 0; i < panjangArray; i++)
    {
        penghitung++;
        if (data[i] == cari)
        {
            ketemu = 1;
            break;
        }
    }
    if (ketemu == 1)
    {
        cout << "Data ada!" << endl;
        cout << "Proses perbandingan terjadi sebanyak : " << penghitung << endl;
    }
    else
    {
        cout << "Data tidak ada!" << endl;
    }
    return 0;
}
```

ScreenShoot Program



```
PS C:\Users\Said\Documents\Basisw\modul11> cd /c:\Users\Said\Documents\Basisw\modul11 ; if ($?) { g++ guided2.cpp -o guided2 } ; if ($?) { ./guided2 }
Masukkan data yang ingin dicari = Avriel
Data ada!
Proses perbandingan terjadi sebanyak : 1
```

Deskripsi Progam

- Panjang aarray 6 dengan tipe data string. Berisi data "Avriel", "Bima", "Yanu", "Arsyad", "Habib", "Zalfa"

- Menjalankan for dari indeks 0 hingga indeks panjangarray
- Disetiap perulangan variabel akan bertambah 1
- Jika data pada i sama dengan yang dicari maka variabel ketemu akan bernilai 1 dan akan berhenti. Jika perulangan selesai dan variabel masih 0, maka data yang dicari tidak ada

UNGUIDED

1. Unguided 1 Source Code

```
#include <iostream>

using namespace std;

int main()
{

    int n;
    int *data;
    int cari;
    bool ketemu;

    cout << "Masukkan panjang array: ";
    cin >> n;

    data = new int[n];
    for (int i = 0; i < n; i++)
    {
        cout << "Masukkan elemen ke-" << i + 1 << ": ";
        cin >> data[i];
    }

    cout << "Masukkan nilai yang ingin dicari: ";
    cin >> cari;

    ketemu = false;
    for (int i = 0; i < n; i++)
    {
        if (data[i] == cari)
        {
            ketemu = true;
            break;
        }
    }

    if (ketemu)
    {
        cout << "Data ditemukan!" << endl;
    }
}
```

```

else
{
    cout << "Data tidak ditemukan!" << endl;
}

delete[] data;

return 0;
}

```

ScreenShoot Program

```

PS C:\Users\Said\Documents\beasiswa\modul11> cd "C:\Users\Said\Documents\beasiswa\modul11" ; if ($?) { g++ unguided1.cpp -o unguided1 } ; if ($?) { .\unguided1 }
Masukkan panjang array: 5
Masukkan elemen ke-1: 2
Masukkan elemen ke-2: 4
Masukkan elemen ke-3: 7
Masukkan elemen ke-4: 2
Masukkan elemen ke-5: 9
Masukkan nilai yang ingin dicari: 9
Data ditemukan!

```

Penjelasan

- Menggunakan tipe data int dan boolean
- Mendeklarasikan variabel n untuk menyimpan panjang array, variabel data untuk menyimpan array, variabel cari menyimpan nilai yang dicari, variabel ketemu untuk menunjukkan apakah nilai yang dicari ditemukan
- Disetiap perulangan variabel akan bertambah 1
- Jika variabel ketemu bernilai true, maka data yang dicari ditemukan, apabila ketemu bernilai false maka data tidak ditemukan

2. Unguided 2

Source Code

```

#include <iostream>

using namespace std;

int main()
{
    int n;
    string *data;
    string cari;
    bool ketemu;

    cout << "Masukkan panjang array: ";
    cin >> n;

    data = new string[n];

```



```

for (int i = 0; i < n; i++)
{
    cout << "Masukkan elemen ke-" << i + 1 << ": ";
    cin >> data[i];
}

cout << "Masukkan nilai yang ingin dicari: ";
cin >> cari;

ketemu = false;
for (int i = 0; i < n; i++)
{
    if (data[i] == cari)
    {
        ketemu = true;
        break;
    }
}

if (ketemu)
{
    cout << "Kata ditemukan!" << endl;
}
else
{
    cout << "Kata tidak ditemukan!" << endl;
}

delete[] data;

return 0;
}

```

ScreenShoot Program

```

PS C:\Users\Said\Documents\Beasiswa\modul11> cd "c:\Users\Said\Documents\Beasiswa\modul11\" ; if ($?) { g++ unguided2.cpp -o unguided2 } ; if ($?) { .\unguided2 }
Masukkan panjang array: 4
Masukkan elemen ke-1: saya
Masukkan elemen ke-2: makan
Masukkan elemen ke-3: bubur
Masukkan elemen ke-4: ayam
Masukkan nilai yang ingin dicari: ayam
Kata ditemukan!

```

Penjelasan

- Menggunakan tipe data int dan boolean dan string
- Mendeklarasikan variabel n untuk menyimpan panjang array, variabel data untuk menyimpan array, variabel cari menyimpan nilai yang dicari, variabel ketemu untuk menunjukkan apakah nilai yang dicari ditemukan

- Disetiap perulangan variabel akan bertambah 1
- Jika variabel ketemu bernilai true, maka data yang dicari ditemukan, apabila ketemu bernilai false maka data tidak ditemukan

BAB IV

KESIMPULAN

Pencarian (searching) merupakan proses yang fundamental dalam pengolahan data. Proses pencarian adalah menemukan nilai (data) tertentu di dalam sekumpulan data yang bertipe sama (baik bertipe dasar atau bertipe bentukan). Sebagai contoh, untuk mengubah (update) data tertentu, langkah pertama yang harus dilakukan adalah mencari keberadaan data tersebut didalam kumpulannya. Jika data yang dicari ditemukan, maka data tersebut dapat diubah nilainya dengan data yang baru. Aktivitas awal yang sama juga dilakukan pada proses penambahan (insert) data baru. Proses penambahan data dimulai dengan mencari apakah data yang akan ditambahkan sudah terdapat di dalam kumpulan. Jika sudah ada dan mengasumsikan tidak boleh ada duplikasi data maka data tersebut tidak perlu ditambahkan, tetapi jika belum ada, maka tambahkan.

Suatu teknik pencarian data dalam array yang akan menelusuri semua elemenelemen array dari awal sampai akhir, dimana data-data tidak perlu diurutkan terlebih dahulu. Kemungkinan terbaik (best case) adalah jika data yang dicari terletak di indeks array terdepan (elemen array pertama) sehingga waktu yang dibutuhkan untuk pencarian data sangat sebentar (minimal). Kemungkinan terburuk (worst case) adalah jika data yang dicari terletak di indeks array terakhir (elemen array terakhir) sehingga waktu yang dibutuhkan untuk pencarian data sangat lama (maksimal).

Secara umum, algoritme pencarian beruntun berjalan lambat. Waktu pencarian sebanding dengan jumlah elemen larik. Misalkan larik berukuran 11 elemen. Maka, pada kasus di mana x tidak terdapat di dalam larik atau x ditemukan pada elemen yang terakhir, kita harus melakukan perbandingan dengan seluruh elemen larik, yang berarti jumlah perbandingan yang terjadi sebanyak n kali. Kita katakan bahwa waktu pencarian dengan algoritme pencarian beruntun sebanding dengan n . Bayangkan bila larik berukuran 100.000 buah elemen, maka kita harus melakukan perbandingan sebanyak 100.000 buah elemen. Andaikan satu operasi perbandingan elemen larik membutuhkan waktu 0.01 detik, maka untuk 100.000 buah perbandingan diperlukan waktu sebesar 1000 detik atau 16,7 menit.