



LEMBAR ASISTENSI
PRAKTIKUM STRUKTUR DATA
LABORATORIUM TEKNIK KOMPUTER
JURUSAN TEKNIK ELEKTRO FAKULTAS TEKNIK
UNIVERSITAS LAMPUNG

Judul Praktikum : SEARCHING
Praktikan : Alvin Reihansyah Makarim (2115061083)
Asisten : Marselina Rahmawati (2015061046)
Zaki Taufiqurrachman (2015061034)

No.	Catatan	Tanggal	Paraf
1.	Asistensi 1 Tujuan percobaan Perbaiki yang dicoret Perbaiki kesimpulan Perbaiki tugas akhir	9 November 2022	
2.	Asistensi 2 Perbaiki penulisan Tambahkan keterangan variabel Perbaiki kesimpulan Perbaiki source code dan output tugas akhir	14 November 2022	
3.	Asistensi 3 Perbaiki source code dan output tugas akhir	15 November 2022	
4.	ACC	15 November 2022	

Bandar Lampung, 15 November 2022

Zaki Taufiqurrachman
NPM. 2015061034

I. JUDUL PERCOBAAN

SEARCHING

II. TUJUAN PERCOBAAN

Adapun tujuan dari percobaan ini adalah sebagai berikut :

1. Mampu memahami penggunaan searching
2. Mampu menggunakan searching

III. TEORI DASAR

3.1 Searching

Searching adalah suatu metode pencarian data untuk menemukan data / informasi yang sedang dicari di dalam sebuah kumpulan data yang memiliki tipe data sama. Pencarian diperlukan untuk mendapatkan informasi / data dari kumpulan data yang belum diketahui. Pencarian dapat dilakukan terhadap data yang secara keseluruhan berada di dalam memori komputer atau penyimpanan eksternal seperti hardisk. Pencarian yang dilakukan terhadap data yang berada dalam komputer dikenal dengan pencarian internal sedangkan pencarian yang dilakukan pada media penyimpanan eksternal disebut pencarian eksternal. Pencarian internal meliputi pencarian sekuensial (sequential search) dan pencarian biner (binary search).

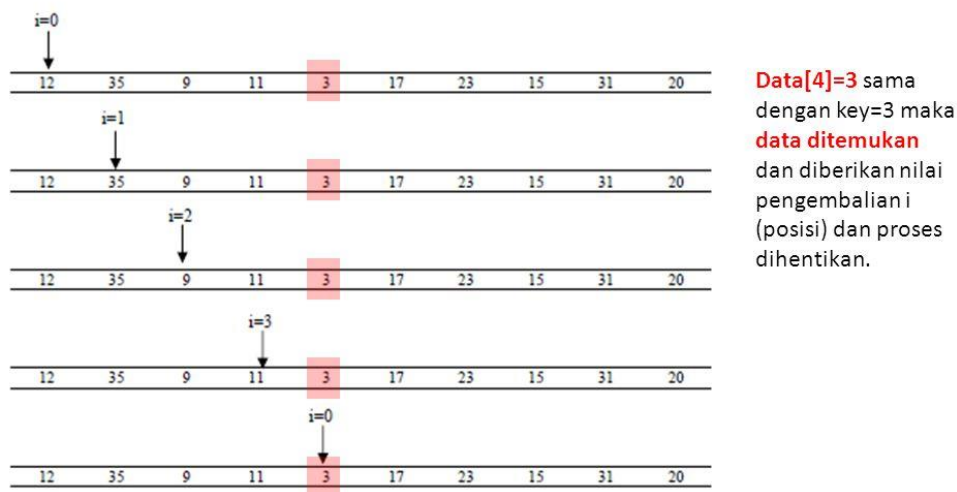
3.2 Sequential searching

Sequential searching adalah suatu teknik pencarian data dalam array (1 dimensi). Sequential searching disebut juga dengan pencarian beruntun. Ini karena sequential searching membandingkan setiap elemen array satu per satu secara beruntun, mulai dari elemen pertama, sampai elemen yang dicari ditemukan atau sampai seluruh

elemen sudah diperiksa. 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).

Ilustrasi Sequential Searching

Mencari posisi data dengan nilai 3. $key=3$;



Gambar 3.2 Ilustrasi sequential searching

Algoritma :

1. Data melakukan perbandingan satu per satu secara berurutan dalam kumpulan data dengan data yang dicari sampai data tersebut ditemukan atau tidak ditemukan.
2. Pada dasarnya, pencarian ini hanya melakukan pengulangan data dari 1 sampai dengan jumlah data (n).
3. Setiap pengulangan, dibandingkan data ke- i dengan data yang sedang dicari.
4. Apabila data sama dengan yang dicari, berarti data telah berhasil ditemukan. Sebaliknya apabila sampai akhir melakukan pengulangan tidak ada data

yang sama dengan yang dicari, berarti data tidak ada yang ditemukan.

Kelebihan :

1. Relatif lebih cepat dan efisien untuk data yang memiliki jumlah terbatas.
2. Algoritma pemrogramannya lebih sederhana

Kekurangan :

1. Kurang cepat untuk melakukan pencarian data dalam jumlah yang besar
2. Beban komputasinya lebih besar

3.3 Binary Searching

Ilustrasi Binary Searching

Contoh Data:

Misalnya data yang dicari **17**

3	9	11	12	15	17	20	23	31	35
awal				tengah		akhir			

Karena $17 > 15$ (data tengah), maka: **awal = tengah + 1**

3	9	11	12	15	17	20	23	31	35
					awal	tengah		akhir	

Karena $17 < 23$ (data tengah), maka: **akhir = tengah - 1**

3	9	11	12	15	17	20	23	31	35
					awal=tengah	akhir			

Karena $17 = 17$ (data tengah), maka **KETEMU!**

Gambar 3.3 Ilustrasi binary searching

Binary Search merupakan sebuah teknik pencarian data dengan cara berulang kali membagi separuh dari jumlah data yang dicari sampai sehingga memperkecil lokasi pencarian menjadi satu data. Dengan teknik ini, kita akan membuang setengah dari jumlah data. Apabila ditemukan kecocokan data maka program akan mengembalikan output, jika tidak pencarian akan terus berlanjut hingga akhir dari pembagian jumlah data tersebut. Algoritma ini biasanya banyak digunakan untuk

mencari di program dengan jumlah data yang banyak, dimana kompleksitas dari algoritma ini adalah $O(\log n)$ di mana n adalah jumlah item. Pada saat menggunakan binary search, data yang berada di dalam array harus diurutkan terlebih dahulu.

Misalkan kita memiliki `int arr[] = {70, 60, 30, 50, 40, 20}`, data pada `int arr` harus diurutkan terlebih dahulu menggunakan teknik sorting seperti bubble sort. Sehingga array kita akan menjadi `int arr[] = {20, 30, 40, 50, 60, 70}`.

Algoritma :

1. Pertama pengambilan data dimulai dari posisi 1 sampai dengan posisi akhir (n)
2. Selanjutnya mencari posisi data yang tengah dengan menggunakan rumus:
 $(\text{posisi awal} + \text{posisi akhir}) \div 2$
3. Setelah itu data yang akan dicari dibandingkan dengan data yang berada di tengah, apakah data tersebut sama atau lebih kecil, atau lebih besar?
4. Seandainya data tersebut lebih besar, maka proses pencarian yang dicari dengan posisi awal adalah posisi tengah + 1
5. Apabila data lebih kecil, maka proses pencarian yang dicari dengan posisi akhir adalah posisi tengah - 1
6. Jika data sama dengan data yang di cari, berarti data tersebut telah ketemu.

Kelebihan :

3. Untuk Pencarian data dalam jumlah yang besar, waktu searching/pencarian lebih cepat karena data telah terurut
4. Beban komputasinya cenderung lebih kecil

Kekurangan :

3. Data harus sudah di-sorting lebih dulu (dalam keadaan terurut) agar lebih mudah dalam pencarian data yang diinginkan.
4. Algoritma pemrogramannya lebih rumit dari sequential search, tidak baik untuk data berangkai

IV. PROSEDUR PERCOBAAN

Adapun source code untuk percobaan ini adalah sebagai berikut :

4.1. Sequential Searching

4.1.1 Percobaan Sequential Searching

```
P3 > Code > C++ sequentialSearching.cpp
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int n = 8;
7      int data[n] = {3, 6, 3, 8, 5, 9, 1, 4};
8      int cari = 4;
9      int i = 0;
10     int flag = 0;
11
12     while(i<n) {
13         if(data[i] == cari) {
14             flag = 1;
15             break;
16         }
17         i++;
18     }
19
20     if(flag == 1)
21     {
22         cout << "ketemu" << endl;
23     }
24     else
25     {
26         cout << "tidak ketemu" << endl;
27     }
28     return 0;
29 }
```

Gambar 4.1.1 Percobaan Sequential Searching

4.1.2 Percobaan Sequential Searching Counter

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int n = 100;
7      int data[n];
8      int cari = 28;
9      int counter = 0;
10     int flag = 0;
11
12     for(int i=0; i < n; i++)
13     {
14         data[i] = rand() % 100 + 1;
15         cout << data[i] << " ";
16     }
17
18     cout << endl;
19
20     for(int i=0; i < n; i++) {
21         if(data[i] == cari) {
22             flag = 1;
23             counter++;
24         }
25     }
26
27     if(flag == 1)
28     {
29         cout << "Ketemu, sebanyak " << counter << endl;
30     }
31     else
32     {
33         cout << "tidak ketemu" << endl;
34     }
35     return 0;
36 }
```

Gambar 4.1.2 Percobaan Sequential Searching Counter

4.1.3 Percobaan Sequential Searching Sentinel

```
P3 > Code > C++ sequentialSearchingSentinel.cpp
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int n = 10;
7      int data[n] = { 7, 5, 3, 8, 5, 9, 7, 4};
8      int cari = 9;
9      int i = 0;
10
11     data[5] = cari;
12
13     while(data[i] != cari) {
14         i++;
15     }
16
17     if(i < 5)
18     {
19         cout << "Ketemu" << endl;
20     }
21     else
22     {
23         cout << "Tidak ketemu" << endl;
24     }
25     return 0;
26 }
```

Gambar 4.1.3 Percobaan Sequential Searching Sentinel

4.2 Binary Searching

4.2.1 Percobaan Binary Searching

```
P3 > Code > C++ binarySearching.cpp
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int n = 10;
6      int data[n] = { 2, 5, 7, 8, 9, 12, 14, 24};
7      int l, r, m;
8      l = 0;
9      r = n-1;
10     int flag = 0;
11     int cari = 12;
12     while(l <= r && flag == 0)
13     {
14         m = (l+r)/2;
15         cout << "data tengah: " << m << endl;
16         if (data[m] == cari)
17         {
18             flag = 1;
19         }
20         else if (cari < data[m])
21         {
22             cout << "cari di kiri" << endl;
23             r = m-1;
24         }
25         else
26         {
27             cout << "cari di kanan" << endl;
28             l = m+1;
29         }
30     }
31
32     if(flag == 1)
33     {
34         cout << "Ketemu" << endl;
35     }
36     else
37     {
38         cout << "Tidak ketemu" << endl;
39     }
40     return 0;
41 }
```

Gambar 4.2.1 Percobaan Binary Searching

4.2.2 Percobaan Binary Searching Interpolation

```
P3 > Code > C++ binarySearchingInterpolation.cpp
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int n = 10;
7      int data[n] = { 2, 15, 27, 38, 39, 42, 54, 74, 81, 105};
8      int low, high;
9      low = 0;
10     high = n-1;
11     int pos;
12     int cari = 81;
13
14     while(cari > data[low] && cari <= data[high]) {
15         pos = (cari-data[low])/(data[high]-data[low]) * (high-low) + low;
16         cout << pos << endl;
17         if(cari > data[pos])
18         {
19             low = pos+1;
20         }
21         else if (cari < data[pos])
22         {
23             high = pos-1;
24         }
25         else
26         {
27             low = pos;
28         }
29     }
30
31     if(cari == data[low])
32     {
33         cout << "Ketemu" << endl;
34     }
35     else
36     {
37         cout << "Tidak ketemu" << endl;
38     }
39
40     return 0;
41 }
```

Gambar 4.2.2 Percobaan Binary Searching Interpolation

V. PEMBAHASAN

Adapun source code percobaan ini adalah sebagai berikut :

5.1. Sequential Searching

5.1.1 Percobaan Sequential Searching

5.1.1.a Source Code Percobaan Sequential Searching

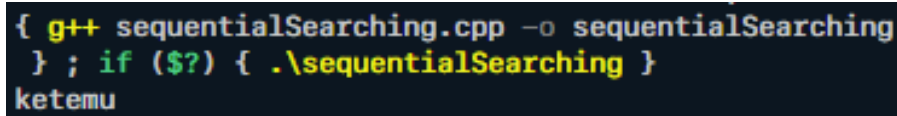
```
P3 > Code > C++ sequentialSearching.cpp
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int n = 8;
7      int data[n] = {3, 6, 3, 8, 5, 9, 1, 4};
8      int cari = 4;
9      int i = 0;
10     int flag = 0;
11
12     while(i<n) {
13         if(data[i] == cari) {
14             flag = 1;
15             break;
16         }
17         i++;
18     }
19
20     if(flag == 1)
21     {
22         cout << "ketemu" << endl;
23     }
24     else
25     {
26         cout << "tidak ketemu" << endl;
27     }
28     return 0;
29 }
```

Gambar 5.1.1.a Source Code Percobaan Sequential Searching

Berdasarkan gambar 5.1.1.a yang merupakan source code dari percobaan sequential searching, dapat dilihat bahwa pada baris ke 1 terdapat penggunaan library iostream ke dalam program yang berfungsi untuk menambahkan perintah input dan output.

Kemudian pada baris ke 2 terdapat instruksi penggunaan penamaan standar untuk compiler. Kemudian pada baris ke 4 terdapat deklarasi fungsi utama dengan tipe data integer `int main()`. Pada baris ke 6 terdapat inisialisasi variabel integer `n` dengan nilai 8. Pada baris ke 7 terdapat inisialisasi array data dengan tipe data integer dan Panjang array `n`. Array tersebut memiliki nilai 3, 6, 3, 8, 5, 9, 1, 4. Pada baris ke 8 terdapat inisialisasi variabel integer `cari` dengan nilai 4. Pada baris ke 9 terdapat inisialisasi variabel integer `i` dengan nilai 0. Pada baris ke 10 terdapat inisialisasi variabel integer `flag` dengan nilai 0. Variabel `flag` ini berfungsi untuk menandakan apakah data yang dicari telah ditemukan atau belum. Pada baris ke 12 terdapat perulangan `while` dengan kondisi selama $i < n$. Di dalam perulangan tersebut, pada baris ke 13, terdapat percabangan `if` dengan kondisi jika `data[i] == cari` dan increment nilai `i` pada baris ke 17. Di dalam percabangan, pada baris ke 14 terdapat inisialisasi nilai dari `flag = 1` yang nantinya akan menandakan bahwa data yang dicari telah ditemukan dan `break` dari perulangan pada baris ke 15. Kemudian pada baris ke 20 terdapat percabangan `if` dengan kondisi jika `flag == 1` tampilkan pesan “ketemu” dan jika tidak maka tampilkan pesan “tidak ketemu”. Kemudian pada baris ke 28 terdapat `return 0` yang menyatakan hasil keluaran dari fungsi `main()` bahwa program berakhir dengan normal.

5.1.1.b Ouput Percobaan Sequential Searching



```
{ g++ sequentialSearching.cpp -o sequentialSearching
} ; if ($?) { .\sequentialSearching }
ketemu
```

Gambar 5.1.1.b Output Percobaan Sequential Searching

Berdasarkan gambar 5.1.1.b yang merupakan output dari percobaan sequential searching, dapat dilihat bahwa program bekerja untuk mencari apakah terdapat data yang diinginkan pada array data. Program menampilkan pesan “ketemu” pada terminal. Ini karena program menemukan data yang dicari. Pada array data, terdapat nomor 3, 6, 3, 8, 5, 9, 1, 4. Dan pada variabel cari, terdapat nomor 4. Pertama program menelusuri elemen pada array secara satu per satu dimulai dari indeks ke 0. Hingga pada indeks ke 7, nilai pada elemen array sama dengan nilai dari variabel cari. Hal itu sesuai dengan percabangan if pada baris ke 13. Karena program menemukan nomor 4 pada array data, program mengubah nilai dari flag yang tadinya 0 menjadi 1, yang mengisyaratkan bahwa data yang dicari telah ditemukan.

5.1.2 Percobaan Sequential Searching Counter

5.1.2.a Source code Percobaan Sequential Searching Counter

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int n = 100;
7      int data[n];
8      int cari = 28;
9      int counter = 0;
10     int flag = 0;
11
12     for(int i=0; i < n; i++)
13     {
14         data[i] = rand() % 100 + 1;
15         cout << data[i] << " ";
16     }
17
18     cout << endl;
19
20     for(int i=0; i < n; i++) {
21         if(data[i] == cari) {
22             flag = 1;
23             counter++;
24         }
25     }
26
27     if(flag == 1)
28     {
29         cout << "Ketemu, sebanyak " << counter << endl;
30     }
31     else
32     {
33         cout << "tidak ketemu" << endl;
34     }
35     return 0;
36 }
```

Gambar 5.1.2.a Source Code Sequential Searching Counter

Berdasarkan gambar 5.1.2.a yang merupakan source code dari sequential searching counter, dapat dilihat bahwa pada baris ke 1 terdapat penggunaan library iostream

ke dalam program yang berfungsi untuk menambahkan perintah input dan output. Kemudian pada baris ke 2 terdapat instruksi penggunaan penamaan standar untuk compiler. Kemudian pada baris ke 4 terdapat deklarasi fungsi utama dengan tipe data integer `int main()`. Pada baris ke 6 terdapat inisialisasi variabel integer `n` dengan nilai 100. Pada baris ke 7 terdapat deklarasi array data dengan tipe data integer dan Panjang array `n`. Pada baris ke 8 terdapat inisialisasi variabel integer `cari` dengan nilai 28. Pada baris ke 9 terdapat inisialisasi nilai dari variabel counter dengan nilai 0 yang berfungsi untuk menyimpan nilai dari berapa kali data yang dicari ditemukan. Pada baris ke 10 terdapat inisialisasi variabel `flag` dengan nilai 0 yang berfungsi untuk menandakan apakah data yang dicari telah ditemukan atau belum. Pada baris ke 12 terdapat perulangan `for` dengan kondisi `int i = 0` dan jika `i < n` maka lakukan increment pada `i`. Di dalam perulangan tersebut, pada baris ke 14, terdapat perintah inisialisasi nilai dari elemen-elemen pada array data dengan cara mengambil angka acak (`random`), mencari sisa baginya dengan 100 ditambah 1. Selain itu pada baris ke 15 terdapat perintah untuk menampilkan nilai dari data yang baru saja diisikan. Pada baris ke 18 terdapat perintah untuk berpindah baris pada terminal. Pada baris ke 20 terdapat perulangan `for` dengan kondisi `int i = 0` dan jika `i < n` maka lakukan increment pada `i`. Di dalam perulangan tersebut, pada baris ke 21, terdapat percabangan `if` dengan kondisi jika `data[i] == cari`. Di dalam percabangan, pada baris ke 22 terdapat inisialisasi nilai dari `flag = 1` dan pada baris ke 23 terdapat increment nilai counter. Kemudian pada baris ke 27 terdapat percabangan `if` dengan kondisi jika `flag == 1` tampilkan pesan “ketemu” dan jika tidak maka tampilkan pesan “tidak ketemu”. Kemudian pada baris ke 35 terdapat `return 0` yang menyatakan hasil keluaran dari fungsi `main()` bahwa program berakhir dengan normal.

5.1.2.b Output Percobaan Sequential Searching Counter

```
chingCounter.cpp -o sequentialSearchingCounter
} ; if ($?) { .\sequentialSearchingCounter }
42 68 35 1 70 25 79 59 63 65 6 46 82 28 62 92
96 43 28 37 92 5 3 54 93 83 22 17 19 96 48 27
72 39 70 13 68 100 36 95 4 12 23 34 74 65 42 1
2 54 69 48 45 63 58 38 60 24 42 30 79 17 36 91
43 89 7 41 43 65 49 47 6 91 30 71 51 7 2 94 4
9 30 24 85 55 57 41 67 77 32 9 45 40 27 24 38
39 19 83 30 42
Ketemu, sebanyak 2
```

Gambar 5.1.2.b Output Percobaan Sequential Searching Counter

Berdasarkan gambar 5.1.2.b yang merupakan output dari percobaan sequential searching counter, dapat dilihat bahwa program bekerja untuk mencari berapa banyak data tersebut ada di dalam array data. Pertama program menampilkan data-data yang telah diinisialisasikan secara acak pada array data dimulai dari indeks ke 0. Hal ini sesuai dengan perintah pada baris ke 15. Kemudian, berdasarkan perintah pada baris ke 20 dan 21, program mencari nilai yang sama antara variabel cari dan nilai elemen dari array data secara satu per satu dari indeks ke 0. Karena pada indeks ke 13 dan 18 nilai tersebut sama, maka terjadi increment sebanyak 2 kali pada variabel counter dan inisialisasi nilai dari flag menjadi 1. Kemudian karena nilai flag = 1, maka sesuai dengan perintah pada baris ke 29, program menampilkan pesan “Ketemu, sebanyak “ yang diikuti dengan nilai dari variabel counter.

5.1.3 Percobaan Sequential Searching Sentinel

5.1.3.a Source Code Percobaan Sequential Searching Sentinel

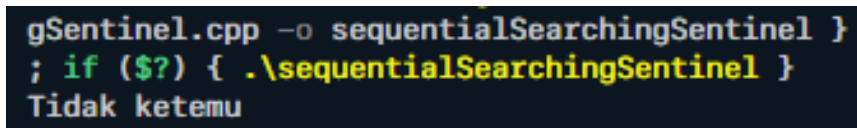
```
P3 > Code > C++ sequentialSearchingSentinel.cpp
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int n = 10;
7      int data[n] = { 7, 5, 3, 8, 5, 9, 7, 4};
8      int cari = 9;
9      int i = 0;
10
11     data[5] = cari;
12
13     while(data[i] != cari) {
14         i++;
15     }
16
17     if(i < 5)
18     {
19         cout << "Ketemu" << endl;
20     }
21     else
22     {
23         cout << "Tidak ketemu" << endl;
24     }
25     return 0;
26 }
```

Gambar 5.1.3.a Source Code Sequential Searching Sentinel

Berdasarkan gambar 5.1.3.a yang merupakan source code dari sequential searching sentinel, dapat dilihat bahwa pada baris ke 1 terdapat penggunaan library iostream ke dalam program yang berfungsi untuk menambahkan perintah input dan output. Kemudian pada baris ke 2 terdapat instruksi penggunaan penamaan standar untuk compiler. Kemudian pada baris ke 4 terdapat deklarasi fungsi utama dengan tipe data integer int main(). Pada baris ke 6 terdapat inisialisasi variabel integer n dengan nilai 10. Pada baris ke 7 terdapat inisialisasi array data dengan tipe data integer dan Panjang array n. Array tersebut memiliki nilai 7, 5, 3, 8, 5, 9, 7, 4. Pada baris ke 8 terdapat inisialisasi variabel integer cari dengan nilai 9. Pada baris ke 9 terdapat inisialisasi variabel integer i dengan nilai 0. Pada baris ke 11 terdapat inisialisasi

nilai array data indeks ke 5 dengan nilai cari. Pada baris ke 13 terdapat perulangan while dengan kondisi jika nilai data[i] tidak sama dengan nilai cari, maka lakukan increment pada nilai i. Pada baris ke 17 terdapat percabangan if dengan kondisi jika $i < 5$ tampilkan pesan “ketemu” dan jika tidak maka tampilkan pesan “tidak ketemu”. Variabel i ini berfungsi sebagai batas pencarian dari data, sehingga program hanya mencari data pada indeks ke 0 hingga 4. Kemudian pada baris ke 25 terdapat return 0 yang menyatakan hasil keluaran dari fungsi main() bahwa program berakhir dengan normal.

5.1.3.b Output Percobaan Sequential Searching Sentinel



```
gSentinel.cpp -o sequentialSearchingSentinel }  
; if ($?) { .\sequentialSearchingSentinel }  
Tidak ketemu
```

Gambar 5.1.3.b Output Percobaan Sequential Searching Sentinel

Berdasarkan gambar 5.1.3.a yang merupakan output dari percobaan sequence searching sentinel, dapat dilihat bahwa program menampilkan pesan “Tidak ketemu”. Ini karena berdasarkan baris ke 11, pencarian dibatasi hanya hingga indeks ke 4 pada array data. Selain itu berdasarkan baris ke 13, perulangan hanya mencari hingga nilai indeks tidak melebihi batas. Indeks 0, 1, 2, 3, dan 4 pada array data, tidak ada yang bernilai 5, sehingga berdasarkan percabangan pada baris ke 17, program menampilkan pesan “Tidak ketemu”.

5.2 Binary Searching

5.2.1 Percobaan Binary Searching

5.2.1.a Source code Percobaan Binary Searching

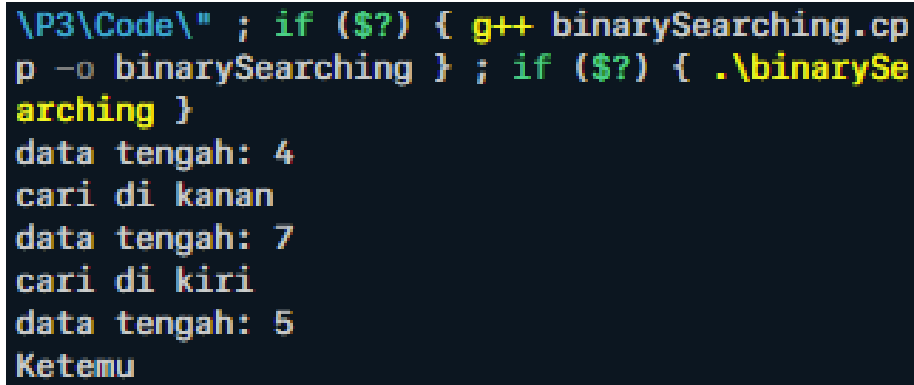
```
P3 > Code > C++ binarySearching.cpp
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int n = 10;
6      int data[n] = { 2, 5, 7, 8, 9, 12, 14, 24};
7      int l, r, m;
8      l = 0;
9      r = n-1;
10     int flag = 0;
11     int cari = 12;
12     while(l <= r && flag == 0)
13     {
14         m = (l+r)/2;
15         cout << "data tengah: " << m << endl;
16         if (data[m] == cari)
17         {
18             flag = 1;
19         }
20         else if (cari < data[m])
21         {
22             cout << "cari di kiri" << endl;
23             r = m-1;
24         }
25         else
26         {
27             cout << "cari di kanan" << endl;
28             l = m+1;
29         }
30     }
31
32     if(flag == 1)
33     {
34         cout << "Ketemu" << endl;
35     }
36     else
37     {
38         cout << "Tidak ketemu" << endl;
39     }
40     return 0;
41 }
```

Gambar 5.2.1 Source Code Percobaan Binary Searching

Berdasarkan gambar 5.2.1.a yang merupakan source code dari binary searching, dapat dilihat bahwa pada baris ke 1 terdapat penggunaan library iostream ke dalam

program yang berfungsi untuk menambahkan perintah input dan output. Kemudian pada baris ke 2 terdapat instruksi penggunaan penamaan standar untuk compiler. Kemudian pada baris ke 3 terdapat deklarasi fungsi utama dengan tipe data integer `int main()`. Pada baris ke 5 terdapat inisialisasi variabel integer `n` dengan nilai 10. Pada baris ke 6 terdapat inisialisasi array data dengan tipe data integer dan Panjang array `n`. Array tersebut memiliki nilai 2, 5, 7, 8, 9, 12, 14, dan 24. Pada baris ke 7 terdapat deklarasi tiga variabel integer dengan nama `l`, `r`, dan `m`. Variabel `l` berfungsi untuk menentukan batas kiri dari data yang dicari, variabel `r` berfungsi untuk menentukan batas kanan dari data yang dicari, dan variabel `m` berfungsi untuk menandakan data tengah dari data yang dicari. Pada baris ke 8 terdapat inisialisasi nilai variabel `l = 0`. Pada baris ke 9 terdapat inisialisasi nilai dari variabel `r = n-1`. Pada baris ke 10 terdapat inisialisasi variabel integer `flag` dengan nilai 0 yang berfungsi sebagai penanda apakah data yang dicari telah ditemukan. Pada baris ke 11 terdapat inisialisasi variabel integer `cari` dengan nilai 12. Pada baris ke 12 terdapat perulangan `while` dengan kondisi jika `l` kurang dari/sama dengan `r` dan `flag = 0`. Di dalam perulangan tersebut, pada baris ke 14, terdapat inisialisasi nilai `m = (l+r) ÷ 2`. Pada baris ke 15 terdapat perintah untuk menampilkan data tengah dan nilai dari variabel `m`. Pada baris ke 16 terdapat percabangan `if` dengan kondisi jika `data[m] = cari`, maka ubah nilai `flag` menjadi 1; jika tidak maka lakukan percabangan `if` dengan kondisi jika `cari < data[m]` maka tampilkan pesan “cari di kiri” dan inisialisasi nilai `r = m-1`; jika tidak, maka tampilkan pesan “cari di kanan” dan inisialisasi nilai `l = m+1`. Kemudian, setelah keluar dari perulangan, pada baris ke 32, terdapat percabangan `if` dengan kondisi jika `i < 5` tampilkan pesan “ketemu” dan jika tidak maka tampilkan pesan “tidak ketemu”. Kemudian pada baris ke 40 terdapat `return 0` yang menyatakan hasil keluaran dari fungsi `main()` bahwa program berakhir dengan normal.

5.2.1.b Output Percobaan Binary Searching



```
\P3\Code\" ; if ($?) { g++ binarySearching.cpp -o binarySearching } ; if ($?) { .\binarySearching }
data tengah: 4
cari di kanan
data tengah: 7
cari di kiri
data tengah: 5
Ketemu
```

Gambar 5.2.1.b Output Percobaan Binary Searching

Berdasarkan gambar 5.2.1.b yang merupakan output dari binary searching, dapat dilihat bahwa pertama program mencari data tengah dengan cara menjumlahkan nomor indeks (bukan nilai) kanan dan kiri array data, lalu dibagi dua. Berdasarkan rumus tersebut, di dapatkan nilai $l=0$, $r = n - 1$, dan $r = 9$. Jika dihitung, maka di dapatkan nilai 4.5, namun komputer membulatkannya ke bawah menjadi 4, sehingga program menampilkan data pada indeks ke 4 dan memasukkannya ke pesan “data tengah : 4”. Kemudian program menampilkan pesan “cari di kanan”. Ini karena tidak memenuhi syarat pada percabangan baris 16, sehingga flag masih bernilai 0. Kemudian program mengecek kondisi pada percabangan berikutnya, yang menghasilkan nilai false, sehingga program menjalankan opsi terakhir dari percabangan tersebut, yaitu menampilkan pesan “cari di kanan” dan mengubah nilai $l = m+1$, yaitu 5. Kemudian, program kembali menghitung nilai tengah dengan nilai $l = 5$, dan $r = 9$. Maka ditemukanlah data tengah berada pada indeks ke 7 dan kemudian menampilkannya ke dalam pesan “data tengah : 7”. Karena syarat pada percabangan baris ke 16 masih belum terpenuhi, maka program menjalankan percabangan kedua, yang menghasilkan nilai true. Kemudian program menghitung ulang kembali nilai tengah dari array tersebut dengan nilai $l = 5$, dan $r = 6$. Maka ditemukanlah data tengah berada pada indeks ke 5. Kali ini, karena nilai tengah sama dengan data yang dicari, maka program mengubah nilai flag menjadi 1 yang menandakan bahwa data yang dicari telah ditemukan dan program menampilkan pesan “Ketemu”.

5.2.2 Percobaan Binary Searching Interpolation

5.2.2.a Source code Percobaan Binary Searching Interpolation

```
P3 > Code > C++ binarySearchingInterpolation.cpp
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int n = 10;
7      int data[n] = { 2, 15, 27, 38, 39, 42, 54, 74, 81, 105};
8      int low, high;
9      low = 0;
10     high = n-1;
11     int pos;
12     int cari = 81;
13
14     while(cari > data[low] && cari <= data[high]) {
15         pos = (cari-data[low])/(data[high]-data[low]) * (high-low) + low;
16         cout << pos << endl;
17         if(cari > data[pos])
18         {
19             low = pos+1;
20         }
21         else if (cari < data[pos])
22         {
23             high = pos-1;
24         }
25         else
26         {
27             low = pos;
28         }
29     }
30
31     if(cari == data[low])
32     {
33         cout << "Ketemu" << endl;
34     }
35     else
36     {
37         cout << "Tidak ketemu" << endl;
38     }
39
40     return 0;
41 }
```

Gambar 5.2.2.a Source Code Percobaan Binary Searching Interpolation

Berdasarkan gambar 5.2.2.a yang merupakan source code dari binary searching interpolation, dapat dilihat bahwa pada baris ke 1 terdapat penggunaan library iostream ke dalam program yang berfungsi untuk menambahkan perintah input dan output. Kemudian pada baris ke 2 terdapat instruksi penggunaan penamaan standar untuk compiler. Kemudian pada baris ke 4 terdapat deklarasi fungsi utama dengan tipe data integer int main(). Pada baris ke 6 terdapat inisialisasi variabel integer n

dengan nilai 10. Pada baris ke 7 terdapat inisialisasi array data dengan tipe data integer dan Panjang array n. Array tersebut memiliki nilai 2, 15, 27, 38, 39, 42, 54, 74, 81, dan 105. Pada baris ke 8 terdapat deklarasi dua variabel integer dengan nama low dan high. Variabel low digunakan untuk menentukan data terendah dan variabel high digunakan untuk menentukan data tertinggi. Pada baris ke 9 terdapat inisialisasi nilai dari variabel low = 0. Pada baris ke 10 terdapat inisialisasi nilai dari variabel high dengan nilai n-1. Pada baris ke 11 terdapat deklarasi variabel integer dengan nama pos yang berfungsi untuk menandai posisi indeks saat sedang melakukan pencarian. Pada baris ke 12 terdapat inisialisasi variabel integer cari dengan nilai 105. Kemudian pada baris ke 14 terdapat perulangan while dengan kondisi jika cari > data[low] dan cari ≤ data[high]. Di dalam perulangan tersebut, pada baris ke 15, terdapat inisialisasi nilai $pos = (cari - data[low]) \div (data[high] - data[low]) \times (high - low) + low$. Pada baris ke 16 terdapat perintah untuk menampilkan nilai dari pos. Pada baris ke 17 terdapat percabangan if dengan kondisi cari > data[pos], maka inisialisasi nilai low = pos+1; jika tidak maka jika cari < data[pos], maka inisialisasi nilai high = pos -1; jika tidak, maka inisialisasi nilai low = pos. Kemudian, setelah keluar dari perulangan, pada baris ke 31, terdapat percabangan if dengan kondisi jika cari = data[low] tampilkan pesan “ketemu” dan jika tidak maka tampilkan pesan “tidak ketemu”. Kemudian pada baris ke 40 terdapat return 0 yang menyatakan hasil keluaran dari fungsi main() bahwa program berakhir dengan normal.

5.2.2.b Output Percobaan Binary Searching Interpolation

```
{ g++ binarySearchingInterpolation.cpp -o
  binarySearchingInterpolation } ; if ($?)
{ .\binarySearchingInterpolation }
0
1
2
3
4
5
6
7
Ketemu
```

Gambar 5.2.2.b Output Percobaan Binary Searching Interpolation

Berdasarkan gambar 5.2.2.b yang merupakan output dari binary searching interpolation, dapat dilihat bahwa program akan mengakses seluruh elemen untuk mencari nilai yang diinginkan user pada variabel cari, yaitu 81. Program mencari data yang sesuai dengan menggunakan rumus $(\text{cari} - \text{data}[\text{low}]) / (\text{data}[\text{high}] - \text{data}[\text{low}]) * (\text{high} - \text{low}) + \text{low}$. Diketahui cari = 81, low = 0, dan high = n-1, yaitu 9. Pada data[low] yang bernilai 0 maka indeks 0 dari data[n] adalah 2 dan nilai high dari data[n] adalah 105. Maka, operasi perhitungan pos ialah $\text{pos} = (81 - 2) / (105 - 2) * (9 - 0) + 0$. Kemudian di dapatkan hasil 0.775 dan oleh komputer dibulatkan ke bawah menjadi 0. Kemudian program mencari nilai pada data[0], yaitu 2, menampilkan nilai variabel pos ke terminal, dan membandingkannya dengan nilai yang dicari. Karena tidak sama dan lebih kecil daripada data yang dicari program akan menjalankan kode program baris 16 yaitu percabangan untuk mendapatkan nilai low, dan pada baris 18 low dapat dihitung menjadi $\text{low} = \text{pos} + 1$ yang dimana pos sudah didapatkan dari perhitungan sebelumnya maka $\text{low} = 0 + 1$ yaitu 1. Kemudian setelah terjadi perubahan data, maka program kembali mencari nilai yang diinginkan dengan rumus $\text{pos} = (81 - 15) / (105 - 15) * (9 - 1) + 1$, sehingga dihasilkan $\text{pos} = 1$. Kemudian program mencari nilai pada data[1], yaitu 15, menampilkan nilai variabel pos ke terminal, dan membandingkannya dengan nilai

yang dicari. Karena tidak sama dan masih lebih kecil daripada data yang dicari program akan menjalankan kode program baris 16 yaitu percabangan untuk mendapatkan nilai low, dan pada baris 18 low dapat dihitung menjadi $low = pos + 1$ yang dimana pos sudah didapatkan dari perhitungan sebelumnya maka $low = 1 + 1$ yaitu 2. Kemudian setelah terjadi perubahan data, maka program kembali mencari nilai yang diinginkan dengan rumus $pos = (81-27)/(105-27)*(9-2)+2$, sehingga dihasilkan $pos = 2$. Kemudian program mencari nilai pada data[2], yaitu 27, menampilkan nilai variabel pos ke terminal, dan membandingkannya dengan nilai yang dicari. Karena tidak sama dan masih lebih kecil daripada data yang dicari program akan menjalankan kode program baris 16 yaitu percabangan untuk mendapatkan nilai low, dan pada baris 18 low dapat dihitung menjadi $low = pos + 1$ yang dimana pos sudah didapatkan dari perhitungan sebelumnya maka $low = 2 + 1$ yaitu 3. Kemudian setelah terjadi perubahan data, maka program kembali mencari nilai yang diinginkan. Program ini akan terus berulang hingga akhirnya ketika low menjadi 8, dan digunakan perhitungan $pos = (81-81)/(105-81)*(9-2)+8$, di dapatkan $pos = 8$. Kemudian program mencari nilai pada data[8], yaitu 81, menampilkan nilai variabel pos ke terminal, dan membandingkannya dengan nilai yang dicari. Karena data yang dicari telah ditemukan, maka program keluar dari perulangan untuk mencari data tersebut, dan berpindah pada percabangan pada baris ke 29. Karena kondisi $cari = data[low]$ terpenuhi, maka program menampilkan pesan “Ketemu” pada terminal.

VI. KESIMPULAN

Adapun kesimpulan dari percobaan ini adalah sebagai berikut :

1. Berdasarkan percobaan 5.1.1 Sequential searching, dapat disimpulkan bahwa variabel flag pada baris ke 10 digunakan sebagai penanda jika data yang dicari telah ditemukan.
2. Berdasarkan percobaan 5.1.2 sequential searching counter, dapat disimpulkan bahwa fungsi variabel counter pada baris ke 9 adalah untuk menghitung berapa banyak data yang sama yang ditemukan pada array.
3. Berdasarkan percobaan 5.1.3 sequential searching sentinel, dapat disimpulkan bahwa pada percobaan ini, program melakukan pembatasan pencarian seperti yang ditunjukkan pada baris ke 11, sehingga program hanya mencari data pada indeks kurang dari 5.
4. Berdasarkan percobaan 5.2.1 binary searching, dapat disimpulkan bahwa pencarian dengan metode binary searching akan memecah list data menjadi dua dan menentukan nilai tengahnya dan membandingkan nilai tengah tersebut dengan data yang dicari hingga menemukannya, seperti yang ada pada baris ke 12 hingga baris ke 30.
5. Berdasarkan percobaan 5.2.1 binary searching, dapat disimpulkan bahwa pencarian data dilakukan dengan cara memeriksa nilai dari indeks yang berada di tengah-tengah data. Ini dapat dilakukan karena sebelum dilakukan pencarian, data diurutkan terlebih dahulu secara ascending. Jika data tidak ditemukan dan nilai yang dicari lebih besar dari nilai yang ditemukan, maka program akan mencari pada indeks berikutnya / nilai yang lebih besar, sedangkan jika nilai yang dicari lebih kecil daripada yang ditemukan, maka program akan mencari pada indeks sebelumnya / nilai yang lebih kecil.

DAFTAR PUSTAKA

- Erliansyah Nasution dan Indra Yatini B.2005. Algoritma dan Struktur Data. Graha Ilmu. Yogyakarta.
- Sjukani, Moh. 2010. (Algoritma Dan Struktur Data I) Dengan C, C++ dan Java. Jakarta: Mitra Wacana Media
- Muhammad, M.A. (018. Struktur Data. Modul Struktur Data Unila PSTI. 2 (2) : 1 - 2.
- Muhammad, Meizano Ardhi. 2018. Modul Praktikum Struktur Data. Lampung Jurusan Teknik Elektro Fakultas Teknik Universitas Lampung.
- Ulum, M. Bahrul Ulum.2018. Modul Praktikum Struktur Data. Jakarta Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Esa Unggul.

TUGAS AKHIR

1. Source code

Adapun source code dari program adalah sebagai berikut :

```
1  #include <iostream>
2  using namespace std;
3
4  void sequentialSearch(int dataInput[], int arrayLength, int search);
5  void binarySearch(int dataInput[], int arrayLength, int search);
6  void inputArrayElement(int dataInput[], int arrayLength);
7  void printArray(int dataInput[], int arrayLength);
8
9  int main()
10 {
11     string choice;
12     int dataInput[100];
13     int arrayLength;
14     int search;
15     bool ongoing = true;
16     bool needChoiceInput = false;
17
18     cout << "===== " << endl;
19     cout << "== Program Penerapan Metode Searching ==" << endl;
20     cout << "===== " << endl;
21
22     while(ongoing){
23         needChoiceInput = true;
24         cout << endl << "===== " << endl;
25         cout << "== Pilih metode pencarian ==" << endl;
26         cout << "===== " << endl;
27         cout << "1. Sequential search" << endl;
28         cout << "2. Binary search" << endl;
29         cout << "3. AKHIRI PROGRAM" << endl;
30
31         while (needChoiceInput){
32             cout << endl << "Silahkan pilih antara [1-3]: ";
33             cin >> choice;
34             cout << "===== " << endl;
35             needChoiceInput = false;
36             switch (choice[0]){
37                 case '1':
38                     cout << "Banyak data : ";
39                     cin >> arrayLength;
40                     dataInput[arrayLength];
41                     inputArrayElement(dataInput, arrayLength);
42                     cout << "Masukkan data yang dicari: ";
43                     cin >> search;
44                     printArray(dataInput, arrayLength);
45                     cout << endl;
46                     sequentialSearch(dataInput, arrayLength, search);
47                     break;
48                 case '2':
49                     cout << "Banyak data : ";
50                     cin >> arrayLength;
51                     dataInput[arrayLength];
52                     inputArrayElement(dataInput, arrayLength);
53                     cout << "Masukkan data yang dicari: ";
```

```

54         cin >> search;
55         printArray(dataInput, arrayLength);
56         cout << endl;
57         binarySearch(dataInput, arrayLength, search);
58         break;
59     case '3':
60         cout << endl << "Program Selesai." << endl;
61         ongoing = false;
62         break;
63     default:
64         cout << "Input tidak tepat." << endl << endl;
65         needChoiceInput = true;
66         break;
67     }
68 }
69 }
70 return 0;
71 }
72
73 void sequentialSearch(int dataInput[], int arrayLength, int search)
74 {
75     int i = 0;
76     int flag = 0;
77
78     while(i < arrayLength) {
79         if(dataInput[i] == search) {
80             flag = 1;
81             break;
82         }
83         i++;
84     }
85
86     if(flag == 1)
87     {
88         cout << "Data " << dataInput[i] << " ditemukan pada indeks ke : " << i << endl;
89     }
90     else
91     {
92         cout << "Data TIDAK ditemukan" << endl;
93     }
94 }
95
96 void binarySearch(int dataInput[], int arrayLength, int search)
97 {
98     int l, r, m;
99     l = 0;
100    r = arrayLength-1;
101    int flag = 0;
102
103    int temporaryArray[arrayLength];
104    for (int i = 0; i < arrayLength; i++){
105        temporaryArray[i] = dataInput[i];
106    };
107
108    cout << "Proses sorting.." << endl;
109
110    for (int i = 0; i < arrayLength-1; i++) {
111        for (int j = 0; j < arrayLength-i-1; j++) {
112            if (temporaryArray[j] > temporaryArray[j+1]) {

```

```

113         int t=temporaryArray[j];
114         temporaryArray[j]=temporaryArray[j+1];
115         temporaryArray[j+1]=t;
116     }
117 }
118 }
119 printArray(temporaryArray, arrayLength);
120
121 while(l <= r && flag == 0)
122 {
123     m = (l+r)/2;
124     cout << endl << "Data tengah: " << m << endl;
125     if (temporaryArray[m] == search)
126     {
127         flag = 1;
128     }
129     else if (search < temporaryArray[m])
130     {
131         cout << "Mencari di kiri..." << endl;
132         r = m-1;
133     }
134     else
135     {
136         cout << "Mencari di kanan..." << endl;
137         l = m+1;
138     }
139 }
140
141 if(flag == 1)
142 {
143     cout << "Data " << temporaryArray[m] << " ditemukan pada indeks ke-" << m << endl;
144 }
145 else
146 {
147     cout << "Data TIDAK ditemukan" << endl;
148 }
149 }
150
151 void inputArrayElement(int dataInput[], int arrayLength)
152 {
153     cout << endl;
154     for(int i = 0; i < arrayLength; i++)
155     {
156         cout << "Masukkan data ke-" << i+1 << ": ";
157         cin >> dataInput[i];
158     }
159     cout << endl;
160 }
161
162 void printArray(int dataInput[], int arrayLength)
163 {
164     cout << endl;
165     cout << " ";
166     for(int i = 0; i < arrayLength; i++)
167     {
168         cout << dataInput[i] << " ";
169     }
170     cout << endl;
171 }

```

Gambar 1 Source code

Berdasarkan gambar 1 yang merupakan source code dari program, dapat dilihat bahwa pada baris ke 1 terdapat penggunaan library iostream ke dalam program yang berfungsi untuk menambahkan perintah input dan output. Kemudian pada baris ke 2 terdapat instruksi penggunaan penamaan standar untuk compiler. Pada baris ke 4

terdapat deklarasi void function dengan nama sequentialSearch dan parameter array integer dataInput, variabel integer arrayLength, dan variabel integer search. Pada baris ke 5 terdapat deklarasi void function dengan nama binarySearch dan parameter array integer dataInput, variabel integer arrayLength, dan variabel integer search. Pada baris ke 6 terdapat deklarasi void function dengan nama inputArrayElement dan parameter array integer dataInput dan variabel integer arrayLength. Pada baris ke 7 terdapat deklarasi void function dengan nama printArray dan parameter array integer dataInput dan variabel integer arrayLength. Kemudian pada baris ke 9 terdapat deklarasi fungsi utama dengan tipe data integer int main(). Di dalam fungsi utama terdapat, pada baris ke 11 terdapat deklarasi variabel string dengan nama choice. Pada baris ke 12 terdapat deklarasi array integer dengan nama dataInput dan panjang array yaitu 100. Pada baris ke 13 terdapat deklarasi dari variabel integer arrayLength. Pada baris ke 14 terdapat deklarasi dari variabel integer search. Pada baris ke 15 terdapat inisialisasi dari variabel boolean ongoing dengan nilai true. Pada baris ke 16 terdapat inisialisasi dari variabel boolean needChoiceInput dengan nilai false. Pada baris ke 18, 19, dan 20 terdapat perintah untuk menampilkan pesan “Program penerapan metode sorting” yang membentuk seperti banner. Pada baris ke 22 terdapat looping while dengan kondisi ongoing hingga baris ke 71. Pada baris ke 23, di dalam looping while, terdapat inisialisasi nilai needChoiceInput dengan nilai true. Pada baris ke 24, 25, dan 26, terdapat perintah untuk menampilkan pesan “Pilih metode pencarian” yang berbentuk seperti banner. Pada baris ke 27 terdapat perintah untuk menampilkan pesan "1. Sequential search". Pada baris ke 28 terdapat perintah untuk menampilkan pesan "2. Binary search". Pada baris ke 29 terdapat perintah untuk menampilkan pesan "3. AKHIRI PROGRAM". Pada baris ke 31 terdapat perulangan while dengan kondisi needChoiceInput hingga baris ke 70. Pada baris 32 di dalam perulangan tersebut, terdapat perintah untuk menampilkan pesan “Silahkan pilih antara [1-3]: “. Pada baris ke 33 terdapat perintah untuk user memasukkan nilai ke variabel choice. Pada baris ke 34 terdapat perintah untuk menampilkan pesan “=====”. Pada baris ke 35, terdapat inisialisasi nilai needChoiceInput menjadi false. Pada baris ke 36 terdapat percabangan switch dengan kondisi choice[0]. Pada baris ke 37 terdapat case 1,

dimana isinya terdapat pada baris ke 38 yaitu perintah untuk menampilkan pesan “Banyak data : “. Pada baris ke 39 terdapat perintah bagi user untuk memasukkan nilai ke variabel `arrayLength`. Pada baris ke 40 terdapat perintah untuk menjadikan nilai dari `arrayLength` sebagai panjang array dari `dataInput`. Pada baris ke 41 terdapat perintah untuk menjalankan fungsi `inputArrayElement` dengan argument `dataInput` dan `arrayLength`. Pada baris ke 42 terdapat perintah untuk menampilkan pesan “Masukkan data yang dicari: “. Pada baris ke 43 terdapat perintah agar user memasukkan nilai ke variabel `search`. Pada baris ke 44 terdapat perintah untuk menjalankan fungsi `printArray` dengan argument `dataInput` dan `arrayLength`. Pada baris ke 46 terdapat perintah untuk menjalankan fungsi `sequentialSearch` dengan argument `dataInput`, `arrayLength`, dan `search`; serta perintah untuk break dari switch pada baris ke 47. Pada baris ke 48 terdapat case 2, dimana isinya terdapat pada baris ke 49 yaitu perintah untuk menampilkan pesan “Banyak data : “. Pada baris ke 50 terdapat perintah bagi user untuk memasukkan nilai ke variabel `arrayLength`. Pada baris ke 51 terdapat perintah untuk menjadikan nilai dari `arrayLength` sebagai panjang array dari `dataInput`. Pada baris ke 52 terdapat perintah untuk menjalankan fungsi `inputArrayElement` dengan argument `dataInput` dan `arrayLength`. Pada baris ke 53 terdapat perintah untuk menampilkan pesan “Masukkan data yang dicari: “. Pada baris ke 54 terdapat perintah agar user memasukkan nilai ke variabel `search`. Pada baris ke 55 terdapat perintah untuk menjalankan fungsi `printArray` dengan argument `dataInput` dan `arrayLength`. Kemudian pada baris ke 57 terdapat perintah untuk menjalankan fungsi `binarySearch` dengan argumen `dataInput`, `arrayLength`, dan `search`; serta perintah untuk break dari switch pada baris ke 58. Pada baris ke 59 terdapat case 3, dimana isinya terdapat pada baris ke 60 yaitu berisi perintah untuk menampilkan pesan “Program selesai”, inisialisasi variabel `ongoing` dengan nilai false pada baris ke 61, dan break dari switch pada baris ke 62. Pada baris ke 63, terdapat default dari switch yang berisi perintah untuk menampilkan pesan “Input tidak tepat” pada baris ke 64, inisialisasi nilai `needChoiceInput` menjadi true pada baris ke 65 dan break dari switch pada baris ke 66. Pada baris ke 70 terdapat `return 0` yang berarti program berjalan dengan normal. Pada baris ke 73 terdapat inisialisasi dari fungsi `void sequentialSearch` dengan parameter array integer `dataInput`, variabel integer `arrayLength`, dan variabel integer `search`. Di dalamnya

yaitu pada baris ke 75 terdapat inisialisasi variabel integer *i* dengan nilai 0. Pada baris ke 76 terdapat inisialisasi variabel integer *flag* dengan nilai 0 yang berfungsi untuk menandakan bahwa data belum ditemukan. Pada baris ke 78 terdapat perulangan *while* dengan kondisi selama *i* < *arrayLength*. Di dalam perulangan tersebut, pada baris ke 79, terdapat percabangan *if* dengan kondisi jika *dataInput[i]* == *search* dan increment nilai *i* pada baris ke 83. Di dalam percabangan, pada baris ke 80 terdapat inisialisasi nilai dari *flag* = 1 yang menandakan bahwa data telah ditemukan dan *break* dari perulangan pada baris ke 81. Kemudian pada baris ke 86 terdapat percabangan *if* dengan kondisi jika *flag* == 1 tampilkan pesan “Data”, *dataInput[i]*, “ditemukan pada indeks ke : ” dan nilai variabel *i*. Kemudian jika *flag* ≠ 1 maka tampilkan pesan “Data TIDAK ditemukan”. Pada baris ke 96 terdapat inisialisasi dari fungsi *void binarySearch* dengan parameter array integer *dataInput*, variabel integer *arrayLength*, dan variabel integer *search*. Di dalamnya, pada baris ke 98 terdapat deklarasi dari tiga variabel integer dengan nama *l*, *r*, dan *m*. Pada baris ke 99 terdapat inisialisasi nilai *l* = 0. Pada baris ke 100 terdapat inisialisasi nilai dari *r* = *arrayLength*-1. Pada baris ke 101 terdapat inisialisasi variabel integer *flag* dengan nilai 0. Pada baris ke 103 terdapat deklarasi dari array integer dengan nama *temporaryArray* dan panjang array sama dengan nilai *arrayLength*. Pada baris ke 104 terdapat perulangan *for* dengan kondisi *int i* = 0 dan *i* < *arrayLength* maka lakukan increment terhadap *i*. Di dalam perulangan tersebut, pada baris ke 105, terdapat inisialisasi nilai *temporaryArray* indeks ke *i* = nilai dari *dataInput* indeks ke *i*. Keluar dari perulangan, pada baris ke 108, terdapat perintah untuk menampilkan pesan “Proses sorting...”. Pada baris ke 110 terdapat perulangan *for* dengan kondisi *int i* = 0 dan jika *i* < *arrayLength* - 1 maka lakukan increment terhadap *i*. Di dalam perulangan tersebut terdapat perulangan (nested loop) *for* dengan kondisi *int j* = 0 dan jika *j* < *arrayLength* - 1 maka lakukan increment terhadap *j*. Di dalam perulangan tersebut, tepatnya pada baris ke 112, terdapat percabangan *if* dengan kondisi *temporaryArray[j]* > *temporaryArray[j+1]*. Di dalam percabangan *if* tersebut, terdapat inisialisasi variabel integer dengan nama *t* dan nilai dari variabel *temporaryArray* indeks ke *j*, inisialisasi nilai *temporaryArray* indeks ke *j* dengan nilai dari indeks berikutnya, dan inisialisasi nilai dari *temporaryArray* indeks ke *j+1* dengan nilai dari *t*. Kemudian setelah keluar dari

nested loop, pada baris ke 119 terdapat perintah untuk menjalankan fungsi `printArray` dengan argument `temporaryArray` dan `arrayLength`. Pada baris ke 121 terdapat perulangan `while` dengan kondisi jika $l \leq r$ dan `flag = 0`. Di dalam perulangan tersebut, pada baris ke 123, terdapat inisialisasi nilai $m = (l+r) \div 2$. Pada baris ke 124 terdapat perintah untuk menampilkan pesan “Data tengah: “ dan nilai dari variabel `m`. Pada baris ke 125 terdapat percabangan `if` dengan kondisi jika `temporaryArray[m] = search`, maka ubah nilai `flag` menjadi 1; jika tidak maka lakukan percabangan `if` dengan kondisi jika `search < temporaryArray[m]` maka tampilkan pesan “Mencari di kiri...” dan inisialisasi nilai $r = m-1$; jika tidak, maka tampilkan pesan “Mencari di kanan...” dan inisialisasi nilai $l = m+1$. Kemudian, setelah keluar dari perulangan, pada baris ke 141, terdapat percabangan `if` dengan kondisi jika `flag = 1` tampilkan pesan “Data ditemukan” dan jika tidak maka tampilkan pesan “Data TIDAK ditemukan”. Pada baris ke 151 terdapat inisialisasi dari fungsi `void inputArrayElement` dengan parameter array integer `dataInput` dan variabel integer `arrayLength`. Pada baris ke 154 terdapat perulangan `for` dengan kondisi $i = 0$ dan jika $i < arrayLength$ maka tampilkan pesan “Masukkan data ke-“, nilai dari $i+1$, dan “: “ pada baris ke 156 dan perintah agar user memasukkan nilai ke array `dataInput` dimulai dari indeks ke 0 hingga ke i pada baris ke 157. Pada baris ke 163 terdapat inisialisasi dari fungsi `void printArray` dengan parameter array integer `dataInput` dan variabel integer `arrayLength`. Di dalamnya, terdapat perintah untuk menampilkan “ “ pada baris ke 165, perulangan `for` pada baris ke 166 dengan kondisi $int\ i = 0$, dan jika $i < arrayLength$ maka lakukan tampilkan nilai dari `dataInput[i]` pada baris ke 168, dan perintah untuk menampilkan baris baru pada baris ke 170.

2. Output

2.1 Output Sequential Search

```
? ) { .\tugasAkhir }  
=====   
== Program Penerapan Metode Searching ==   
=====   
  
=====   
== Pilih metode pencarian ==   
=====   
1. Sequential search  
2. Binary search  
3. AKHIRI PROGRAM  
  
Silahkan pilih antara [1-3]: 1  
=====   
Banyak data : 5  
  
Masukkan data ke-1: 12  
Masukkan data ke-2: 90  
Masukkan data ke-3: 34  
Masukkan data ke-4: 78  
Masukkan data ke-5: 56  
  
Masukkan data yang dicari: 78  
  
12 90 34 78 56  
  
Data 78 ditemukan pada indeks ke : 3  
  
=====   
== Pilih metode pencarian ==   
=====   
1. Sequential search  
2. Binary search  
3. AKHIRI PROGRAM  
  
Silahkan pilih antara [1-3]: 3  
=====   
Program Selesai.
```

Gambar 2.1 Output sequential search

Berdasarkan gambar 2.1 yang merupakan output dari menu sequential search, dapat dilihat bahwa pertama program menampilkan pesan “Program Penerapan Metode Searching” yang berbentuk seperti banner. Ini sesuai dengan perintah pada baris ke 18 hingga 20. Berikutnya program akan menampilkan pesan “Pilih metode pencarian” yang berbentuk seperti banner, sesuai dengan perintah pada baris ke 24 hingga 26. Berikutnya program menampilkan menu yang ada, diantaranya sequential search, binary search, atau akhiri program. Ini sesuai dengan perintah pada baris ke 27 hingga 29. Kemudian program meminta user untuk memilih antara

menu 1 hingga 3 dan meminta user untuk memasukkan nilai. Nilai tersebut nantinya akan dimasukkan ke dalam variabel choice dan menjadi penanda metode searching yang dipilih. Pada percobaan ini, user memilih nilai 1, yaitu sequential search. Berikutnya program menampilkan pesan “Banyak data : “ dan meminta user untuk memasukkan sebuah nilai. Ini sesuai dengan perintah pada baris ke 38 dan 39. Nilai yang diinputkan user nantinya akan dimasukkan ke dalam variabel arrayLength dan menjadi penanda jumlah data yang akan dioperasikan. Kemudian program meminta user untuk memasukkan data satu per satu. Ini sesuai dengan perintah pada baris ke 41 dimana program memanggil fungsi inputArrayElement. Nilai yang diinputkan nantinya akan menjadi data-data yang akan dilakukan operasi pencarian. Pada percobaan ini, saya memasukkan nilai 12, 90, 34, 78, dan 56. Kemudian program menampilkan pesan “Masukkan data yang dicari: “ dan meminta user untuk memasukkan nilai. Ini sesuai dengan perintah pada baris ke 42 dan 43. Nilai yang dimasukkan user akan dimasukkan ke dalam variabel search dan akan menjadi nilai yang dicari pada array data. Pada percobaan ini, saya memasukkan nilai 78. Kemudian program akan menampilkan nilai-nilai tiap elemen pada array dataInput, sesuai dengan perintah pemanggilan fungsi printArray pada baris ke 44. Kemudian program mencari data pada array dataInput secara satu per satu mulai dari indeks ke 0 hingga indeks dimana data tersebut ditemukan atau indeks terakhir. Pada indeks ke 3, program menemukan data yang dicari, sehingga program menghentikan pencarian dan mengubah nilai flag menjadi 1 yang menandakan bahwa data telah ditemukan. Maka, berdasarkan percabangan pada baris ke 86, program menampilkan pesan “Data 78 ditemukan pada indeks ke : 3”. Kemudian program menampilkan ulang menu. Karena kita sudah selesai mencari data, maka kita pilih menu 3 yang akan menampilkan pesan “Program selesai” yang sesuai dengan perintah pada baris ke 60.

2.2 Output Binary Search

```
? ) { .\tugasAkhir }
=====
== Program Penerapan Metode Searching ==
=====

=====
== Pilih metode pencarian ==
=====
1. Sequential search
2. Binary search
3. AKHIRI PROGRAM

Silahkan pilih antara [1-3]: 2
=====
Banyak data : 5

Masukkan data ke-1: 12
Masukkan data ke-2: 90
Masukkan data ke-3: 34
Masukkan data ke-4: 78
Masukkan data ke-5: 56

Masukkan data yang dicari: 78

    12 90 34 78 56

Proses sorting...

    12 34 56 78 90

Data tengah: 2
Mencari di kanan...

Data tengah: 3
Data 78 ditemukan pada indeks ke-3

=====
== Pilih metode pencarian ==
=====
1. Sequential search
2. Binary search
3. AKHIRI PROGRAM

Silahkan pilih antara [1-3]: 3
=====

Program Selesai.
```

Gambar 2.2 Output Binary Search

Berdasarkan gambar 2.2 yang merupakan output dari menu binary search, dapat dilihat bahwa pertama program menampilkan pesan “Program Penerapan Metode Searching” yang berbentuk seperti banner. Ini sesuai dengan perintah pada baris ke 18 hingga 20. Berikutnya program akan menampilkan pesan “Pilih metode

pencarian” yang berbentuk seperti banner, sesuai dengan perintah pada baris ke 24 hingga 26. Berikutnya program menampilkan menu yang ada, diantaranya sequential search, binary search, atau akhiri program. Ini sesuai dengan perintah pada baris ke 27 hingga 29. Kemudian program meminta user untuk memilih antara menu 1 hingga 3 dan meminta user untuk memasukkan nilai. Nilai tersebut nantinya akan dimasukkan ke dalam variabel choice dan menjadi penanda metode searching yang dipilih. Karena kita akan mencoba binary search, maka dipilihlah menu nomor 2. Berikutnya program menampilkan pesan “Banyak data : “ dan meminta user untuk memasukkan sebuah nilai. Ini sesuai dengan perintah pada baris ke 49 dan 50. Nilai yang diinputkan user nantinya akan dimasukkan ke dalam variabel arrayLength dan menjadi penanda jumlah data yang akan dioperasikan. Kemudian program meminta user untuk memasukkan data satu per satu. Ini sesuai dengan perintah pada baris ke 52 dimana program memanggil fungsi inputArrayElement. Nilai yang diinputkan nantinya akan menjadi data-data yang akan dilakukan operasi pencarian. Pada percobaan ini, saya memasukkan nilai 12, 90, 34, 78, dan 56. Kemudian program menampilkan pesan “Masukkan data yang dicari: “ dan meminta user untuk memasukkan nilai. Ini sesuai dengan perintah pada baris ke 53 dan 54. Nilai yang dimasukkan user akan dimasukkan ke dalam variabel search dan akan menjadi nilai yang dicari pada array data. Pada percobaan ini, saya memasukkan nilai 78. Kemudian program akan menampilkan nilai-nilai tiap elemen pada array dataInput, sesuai dengan perintah pada baris ke 55. Kemudian program menampilkan pesan “Proses sorting...”. Ini karena pada binary search, kita perlu melakukan sorting terhadap datanya terlebih dahulu sebelum dilakukan operasi pencarian pada data tersebut. Kemudian program menampilkan data yang telah diurutkan. Pada metoda binary search, program membandingkan nilai yang dicari dengan nilai tengah dari data. Program mencari data tengah dengan rumus $m = (l+r)/2$, dimana m adalah data tengah, l adalah indeks data paling kiri, dan r adalah indeks data paling kanan. Di dapatkan nilai $l = 0$ dan $r = \text{arrayLength} - 1$, yaitu 4. Kemudian pada perhitungan $(0 + 4) / 2$, di dapatkan nilai 2. Pada indeks tersebut, nilai data tengah yang ditemukan adalah 56. Karena tidak sama dengan data yang dicari, maka program mengubah nilai $l = m+1$, yaitu 3. Karena data yang dicari $>$ data yang ditemukan, maka program mencari ke kanan, sesuai dengan percabangan

pada baris ke 134. Kemudian program melakukan perhitungan $(3 + 4) / 2 = 3.5$ dan komputer membulatkannya menjadi 3. Pada indeks 3, nilai data tengah yang ditemukan adalah 78. Karena data yang ditemukan = data yang dicari, maka program mengubah nilai flag menjadi 1 yang berarti bahwa data telah ditemukan. Karena flag = 1, maka kondisi dari perulangan untuk mencari nilai sudah tidak memenuhi, sehingga program keluar dari perulangan tersebut. Kemudian pada percabangan pada baris ke 141, karena flag = 1, maka program menampilkan pesan “Data 78 ditemukan pada indeks ke-3”. Kemudian program menampilkan ulang menu. Karena kita sudah selesai mencari data, maka kita pilih menu 3 yang akan menampilkan pesan “Program selesai” yang sesuai dengan perintah pada baris ke 60.