

LAPORAN AKHIR

ALGORITMA DAN PEMROGRAMAN

LAPORAN KE-12



Disusun Oleh:

Nama: Andri Firman Saputra

NIM : 201011402125

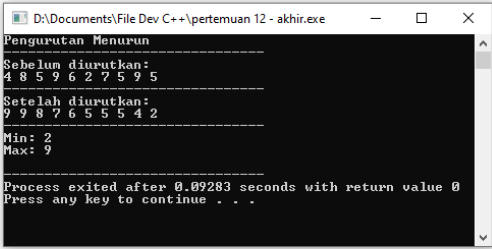
Kelas : 02TPLP023 – Pagi

TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PAMULANG

Jl. Surya Kencana No. 1 Pamulang Telp (021)7412566, Fax. (021)7412566
Tangerang Selatan - Banten

Tugas Akhir – Pertemuan 12

```
pertemuan 12 - akhir.cpp
1  #include<iostream>
2  using namespace std;
3
4  void BubbleSort(int data[], int n);
5
6  int main()
7  {
8      int i,
9      n = 9,
10     data[] = {4, 8, 5, 9, 6, 2, 7, 5, 9, 5};
11
12     cout<<"Pengurutan Menurun"<<endl;
13     cout<<"-----"<<endl;
14     cout<<"Sebelum diurutkan: "<<endl;
15
16     for(i = 0; i <= n; i++)
17     {
18         cout<<data[i]<<" ";
19     }
20
21     cout<<endl;
22
23     cout<<"-----"<<endl;
24
25     BubbleSort(data, n);
26
27     cout<<"Setelah diurutkan: "<<endl;
28
29     for(i = 0; i <= n; i++)
30     {
31         cout<<data[i]<<" ";
32     }
33
34     cout<<endl;
35     cout<<"-----"<<endl;
36
37     cout<<"Min: "<<data[9]<<endl;
38     cout<<"Max: "<<data[0]<<endl;
39
40     return 0;
41 }
42
43 void BubbleSort(int array1[], int n)
44 {
45     int i, j, tmp;
46
47     for(i = 1; i <= n; i++)
48     {
49         for(j = n; j >= i; j--)
50         {
51             if(array1[j] > array1[j-1])
52             {
53                 tmp = array1[j];
54                 array1[j] = array1[j-1];
55                 array1[j-1] = tmp;
56             }
57         }
58     }
59 }
60
61 }
```



```
Pengurutan Menurun
-----
Sebelum diurutkan:
4 8 5 9 6 2 7 5 9 5
-----
Setelah diurutkan:
9 9 8 7 6 5 5 4 2
-----
Min: 2
Max: 9
-----
Process exited after 0.09283 seconds with return value 0
Press any key to continue . . .
```

Source Code:

```
#include<iostream>

using namespace std;

void BubbleSort(int data[], int n);

int main()
{
    int i,
    n = 9,
    data[] = {4, 8, 5, 9, 6, 2, 7, 5, 9, 5};

    cout<<"Pengurutan Menurun"<<endl;
    cout<<"-----"<<endl;
    cout<<"Sebelum diurutkan: "<<endl;

    for(i = 0; i <= n; i++)
    {
        cout<<data[i]<<" ";
    }

    cout<<endl;

    cout<<"-----"<<endl;

    BubbleSort(data, n);

    cout<<"Setelah diurutkan: "<<endl;

    for(i = 0; i <= n; i++)
    {
        cout<<data[i]<<" ";
    }

    cout<<endl;
```

```

        cout<<"-----"<<endl;

        cout<<"Min: "<<data[9]<<endl;
        cout<<"Max: "<<data[0]<<endl;

        return 0;
    }

void BubbleSort(int array1[], int n)
{
    int i, j, tmp;

    for(i = 1; i <= n; i++)
    {
        for(j = n; j >= i; j--)
        {
            if(array1[j] > array1[j-1])
            {
                tmp = array1[j];

                array1[j] = array1[j-1];

                array1[j-1] = tmp;
            }
        }
    }
}

```

Kesimpulan:

Pada pertemuan ke 12 ini saya dapat menarik kesimpulan, Sorting adalah suatu proses pengurutan data yang sebelumnya disusun secara acak atau tidak teratur menjadi urut dan teratur menjadi urut dan teratur menurut suatu aturan tertentu. Biasanya pengurutan terbagi menjadi 2 yaitu : ascending (pengurutan dari karakter/ angka kecil ke karakter / angka besar ke karakter/ angka kecil) Ada banyak cara yang dapat dilakukan untuk melakukan proses pengurutan dari paling atas ke paling bawah atau sebaliknya. Untuk melakukan proses pengurutan dapat menggunakan beberapa metode antara lain:

1) Bubble sort

Bubble sort adalah suatu metode pengurutan yang membandingkan elemen yang sekarang dengan elemen berikutnya. Perbandingan alamatnya dapat dimulai dari data yang paling awal atau yang paling akhir. Apabila elemen yang sekarang (sebelumnya) lebih besar dari elemen berikutnya, maka posisi di tukar, kalau tidak posisinya tetap atau tidak perlu ditukar.

2) Selection sort

Selection sort adalah suatu metode pengurutan yang membandingkan elemen yang sekarang dengan elemen berikutnya sampai elemen terakhir. Jika ditemukan elemen lain yang lebih kecil dari elemen sekarang maka di catat posisinya dan langsung ditukar.

3) Quick sort

Quick sort adalah suatu metode pengurutan yang membandingkan suatu elemen (pivot) dengan elemen yang lain dan menyusun sedemikian rupa sehingga elemen yang lain lebih kecil dari pada pivot terletak disebelah kiri pivot sedangkan elemen yang lebih besar dari pivot diletakkan disebelah kanan pivot.

4) Merge sort

Merge sort adalah suatu metode pengurutan yang membandingkan elemen satu dengan elemen yang lain, apabila nilainya lebih kecil maka datanya ditampung di elemen yang lain lagi.

LAPORAN AWAL

ALGORITMA DAN PEMROGRAMAN

LAPORAN KE-13



Disusun Oleh:

Nama: Andri Firman Saputra

NIM : 201011402125

Kelas : 02TPLP023 – Pagi

TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PAMULANG

Jl. Surya Kencana No. 1 Pamulang Telp (021)7412566, Fax. (021)7412566
Tangerang Selatan - Banten

Nama : Andri Firman Saputra
NIM : 201011402125

Praktikum Algoritma
Laporan Awal - Perkenalan 13

Teori Dasar

a) Pendahuluan

Proses pencarian suatu elemen di dalam array disebut searching, ada 2 macam pencarian yaitu pencarian sekuensial (sequential searching) dan pencarian biner (binary searching).

Perbedaannya terletak pada keadaan suatu elemen atau data yg berada pada array. Pencarian sekuensial digunakan apabila data dalam keadaan acak atau tidak urut.

Sedangkan pencarian biner digunakan pada data yg sudah dalam keadaan urut.

b) Pencarian Sekuensial

Pencarian sekuensial menggunakan prinsip sebagai berikut: data yg ada pada suatu array dibandingkan satu persatu dgn data yg dicari.

Pencarian ini dilakukan dgn melakukan suatu pengulangan dari 1 sampai semua data yg ada. Pada setiap kali pengulangan, dibandingkan data yg posisinya ke - i dengan data yg dicari atau dimausud. Apabila sama, maka data tersebut telah ditemukan dan proses pengulangan dihentikan.

Sebaliknya, kalau sampai pengulangan selesai dan data yg dicari tidak ditemukan, maka tsb tidak ada.

Tugas Pendahuluan

1. Apa yg dimaksud dgn Searching?

Searching adalah proses pencarian suatu elemen di dalam array.

2. Jelaskan kelebihan dan kekurangan Searching pada bahasa C/C++!

Kelebihan: apabila data yg dicari letaknya pada data-data awal sehingga prosesnya berjalan cepat.

Kekurangan: apabila data yg dicari letaknya pada data terakhir maka dalam penggunaan waktu, proses ini berjalan lama.

3. Sebagai Programmer, mengapa anda menggunakan Searching? Karena dapat memudahkan pencarian data.



4. Buatlah contoh algoritma dan program sederhana menggunakan Searching dgn memauai flowchart!

Jawaban No. 4

```
search.cpp
1 #include<iostream>
2 using namespace std;
3
4 void BinSearch(int data[], int n, int x, int *idx);
5
6 int main()
7 {
8     int data[] = {1, 16, 25, 30, 45, 55, 68, 75, 82, 93};
9     int idx, x, i, jmlData = 10;
10
11     cout<<"Elemen Array: ";
12     for(i = 0; i < jmlData; i++)
13     {
14         cout<<data[i]<<" ";
15     }
16     cout<<endl;
17
18     cout<<"Masukkan data yang akan dicari: ";
19     cin>>x;
20
21     BinSearch(data, jmlData, x, &idx);
22     if(idx != -1)
23     {
24         cout<<"Data yang dicari berada pada indeks: "<<idx<<endl;
25     }
26     else
27     {
28         cout<<"Data yang dicari tidak ada dalam array"<<endl;
29     }
30 }
31
32 void BinSearch(int data[], int n, int x, int *idx)
33 {
34     bool ketemu = false;
35     int top = n-1, bottom = 0, mid;
36     int i = 0;
37     while(bottom <= top && !ketemu)
38     {
39         mid = (top + bottom) / 2;
40
41         if(data[mid] == x)
42         {
43             ketemu = true;
44         }
45         else
46         {
47             if(data[mid] > x)
48             {
49                 top = mid - 1;
50             }
51             else
52             {
53                 bottom = mid + 1;
54             }
55         }
56     }
57
58     if(ketemu)
59     {
60         *idx = mid;
61     }
62     else
63     {
64         *idx = -1;
65     }
66 }
67 }
```



The image shows three screenshots of the search.exe application output, demonstrating the binary search process for different input values.

Screenshot 1: The application displays the array elements: 1 16 25 30 45 55 68 75 82 93. It prompts the user to enter a search value. The user enters 16. The output shows: "Data yang dicari berada pada indeks: 1". The process exits after 2.901 seconds with return value 0.

Screenshot 2: The application displays the array elements: 1 16 25 30 45 55 68 75 82 93. It prompts the user to enter a search value. The user enters 75. The output shows: "Data yang dicari berada pada indeks: 7". The process exits after 2.546 seconds with return value 0.

Screenshot 3: The application displays the array elements: 1 16 25 30 45 55 68 75 82 93. It prompts the user to enter a search value. The user enters 70. The output shows: "Data yang dicari tidak ada dalam array". The process exits after 4.04 seconds with return value 0.

Source Code:

```
#include<iostream>

using namespace std;

void BinSearch(int data[], int n, int x, int *idx);

int main()
{
    int data[] = {1, 16, 25, 30, 45, 55, 68, 75, 82, 93};
    int idx, x, i, jmlData = 10;

    cout<<"Elemen Array: ";
    for(i = 0; i < jmlData; i++)
    {
        cout<<data[i]<<" ";
    }
    cout<<endl;

    cout<<"Masukkan data yang akan dicari: ";
    cin>>x;

    BinSearch(data, jmlData, x, &idx);
    if(idx != -1)
    {
        cout<<"Data yang dicari berada pada indeks: "<<idx<<endl;
    }
    else
    {
        cout<<"Data yang dicari tidak ada dalam array"<<endl;
    }
}

void BinSearch(int data[], int n, int x, int *idx)
{
```

```

bool ketemu = false;
int top = n-1, bottom = 0, mid;
int i = 0;
while(bottom <= top && !ketemu)
{
    mid = (top + bottom) / 2;

    if(data[mid] == x)
    {
        ketemu = true;
    }
    else
    {
        if(data[mid] > x)
        {
            top = mid - 1;
        }
        else
        {
            bottom = mid + 1;
        }
    }
}

if(ketemu)
{
    *idx = mid;
}
else
{
    *idx = -1;
}
}

```