LAPORAN PRAKTIKUM

Matakuliah	Struktur Data
Pertemuan ke	6
Nama Praktikan	Wijayanto Agung Wibowo
NIM	22.11.4552
NILAI (diisi oleh dosen / asisten praktikum)	

A. Tujuan

Setelah praktikum ini, praktikan diharapkan dapat:

1. Memahami tipe data dasar
2. Memahami tipe data bentukan
3. Stuktur Program menggunakan bahasa C++ Praktikum

B. Hasil Percobaan

1. Percobaan 1

a) Tampilan Coding

```cpp
#include <iostream>
using namespace std;

int main()
{
    int x = 1; //nilai yang akan dicari
    int arr[]{ 5,9,2,7,8,1,6 };

    //nim nama
    cout << "NIM : 22.11.4552" <<endl;
    cout << "Nama : Wijayanto Agung Wibowo" << endl << endl;

    //proses pencarian secara linier
    bool found{ false };
    for (int i = 0; i < 7; i++) {
        if(x == arr[i])
        {
            found = true;
            cout << "Data ditemukan di indeks ke-" << i;
            break;
        }
    }
}
```

b) Hasil Running

```
Nama : Wijayanto Agung Wibowo
NIM : 22.11.4552

Data ditemukan di index ke- 5
--------------------------------
Process exited after 0.05179 seconds with return value 0
Press any key to continue . . .
```

c) Penjelasan

Linear search menggunakan Teknik for, untuk mencari data yang kita inginkan. Data di cek dari index ke 0 sampai indeks ke 7. Apabila data yang kita cari ditemukan, maka i nya itu ditampilkan sebagai informasi indeks data yang kita cari.

2. Percobaan 2

a) Tampilan Coding

```cpp
#include <iostream>
using namespace std;

int main() {
    //nim nama
    cout << "NIM : 22.11.4552" << endl;
    cout << "Nama : Wijayanto Agung Wibowo" << endl << endl;

    //x adalah nilai yang akan kita cari
    int m, x = 1;
    int arr[]{ 1, 2, 5, 7, 8, 9, 9, 11, 15, 16, 20 };
    int l{ 0 };
    int r{ 10 };

    //proses pencarian binary search
    bool found{ false };
    while (l <= r && !found)
    {
        m = l + (r - 1) / 2;

        if (x == arr[m]) {
            found = true;
            cout << "Data ditemukan di index ke-" << m;
        }
        if (x < arr[m]) r = m - 1;
        else if (x > arr[m]) l = m + 1;

    }

}
```

b) Hasil Running

```
Nama : Wijayanto Agung Wibowo
NIM : 22.11.4552

Data ditemukan di indeks ke- 2
--------------------------------
Process exited after 0.05298 seconds with return value 0
Press any key to continue . . .
```

c) Penjelasan

Binary search adalah mencari data indeks dengan cara menentukan nilai tengah dari indeks dimana datanya sudah diurut. Pencarian data nya yaitu indeks awal – indeks akhir / 2.

Apabila nilai indeks tengah merupakan nilai yang dicari, maka akan menampilkan nilai n. jika tidak, jika nilai tengah lebih dari data yang dicari, maka indeks akhir menjadi nilai tengah – 1. Jika nilai tengah lebih dari nilai tengah, maka indeks akhir menjadi nilai tengah + 1.

3. Percobaan 3

a) Tampilan coding

```
#include <iostream>
#include <conio.h>
using namespace std;

int data1[10], data2[10];
int n;

int tukar(int a, int b) {
    int t;
    t = data1[b];
    data1[b] = data1[a];
    data1[a] = t;
    return 0;
}

int input() {
    cout << "Masukan Jumlah Data = ";
    cin >> n;

    cout << endl;

    for (int i = 0; i < n; i++) {
        cout << "Masukan data ke -" << i + 1 << " = ";
        cin >> data1[i];

        data2[i] = data1[i];
    }
    cout << endl;
    return 0;
}

int tampil() {
    for (int i = 0; i < n; i++) {
        cout << "[" << data1[i] << "] ";
    }
    cout << endl;
    return 0;
}

```

```
39
40  int buuble_sort() {
41      for (int i = 1; i < n; i++) {
42          for (int j = n - 1; j >= i; j--) {
43              if (data1[j] < data1[j - 1]) {
44                  tukar(j, j - 1);
45              }
46          }
47          tampil();
48      }
49      cout << endl;
50      return 0;
51  }
52
53  int main() {
54      cout << "ALGORITMA BUUBLE SORT" << endl;
55      cout << "-------------------------" << endl;
56      input();
57      cout << "Proses buuble sort" << endl;
58      tampil();
59      buuble_sort();
60      return 0;
61  }
```

b) Hasil running

```
Nama : Wijayanto Agung Wibowo
NIM : 22.11.4552

ALGORITMA BUBBLE SORT
----------------------
Masukan Jumlah Data = 4

Masukan Data Ke-1 = 34
Masukan Data Ke-2 = 57
Masukan Data Ke-3 = 54
Masukan Data Ke-4 = 32

Proses Buuble Sort
[34] [57] [54] [32]
[32] [34] [57] [54]
[32] [34] [54] [57]
[32] [34] [54] [57]
```

c) Penjelasan

Penjelasan fungsi:

- Tukar: inputkan data a dan b. lalu variable b masukan ke variable t, variable a masukan ke variable b, varible t masukan ke variable a
- Input: Menggunakan Teknik for, dimana kita memasukan data berasarkan jumlah nilai yang kita inputkan. Lalu masukan ke variable data array
- Tampil: Menggunakan Teknik for, menampilkan Semua data array ke n sampai Panjang array yang terakhir

- Buuble sort : Membandingkan dua elemen adjacent dalam array dan menukarnya jika urutannya salah. Algoritma ini terus melintasi array hingga tidak ada lagi elemen yang perlu ditukar, menunjukkan bahwa array sudah terurut.

4. Percobaan 4
    a) Tampilan coding

```cpp
#include <iostream>
#include <conio.h>
using namespace std;

int data1[10], data2[10];
int n;

int tukar(int a, int b) {
    int t;
    t = data1[b];
    data1[b] = data1[a];
    data1[a] = t;
    return 0;
}

int input() {
    cout << "Masukan Jumlah Data = ";
    cin >> n;

    cout << endl;

    for (int i = 0; i < n; i++) {
        cout << "Masukan data ke -" << i + 1 << " = ";
        cin >> data1[i];

        data2[i] = data1[i];
    }
    cout << endl;
    return 0;
}

int tampil() {
    for (int i = 0; i < n; i++) {
        cout << "[" << data1[i] << "] ";
    }
    cout << endl;
    return 0;
}

int buuble_sort() {
    for (int i = 1; i < n; i++) {
        for (int j = n - 1; j >= i; j--) {
            if (data1[j] < data1[j - 1]) {
                tukar(j, j - 1);
            }
        }
        tampil();
    }
    cout << endl;
    return 0;
}
```

```
52
53  int selection_sort(int data1[], int n) {
54      int i, j, posisi, tukar;
55      for (i = 0; i < (n - 1); i++) {
56          posisi = i;
57          for (j = i + 1; j < n; j++) {
58              if (data1[posisi] > data1[j]) {
59                  posisi = j;
60              }
61          }
62          if (posisi != i) {
63              tukar = data1[i];
64              data1[i] = data1[posisi];
65              data1[posisi] = tukar;
66          }
67
68      }
69      return 0;
70  }
71
72  int main() {
73      //nim nama
74      cout << "Nama : Wijayanto Agung Wibowo" << endl;
75      cout << "NIM : 22.11.4552" << endl << endl;
76
77      cout << "ALGORITMA BUUBLE SORT" << endl;
78      cout << "-------------------------" << endl;
79      input();
80      cout << "Proses buuble sort" << endl;
81      tampil();
82      buuble_sort();
83      return 0;
84  }
```

b) Hasil running

```
Nama : Wijayanto Agung Wibowo
NIM : 22.11.4552

ALGORITMA SELECTION SORT
----------------------
Masukan Jumlah Data = 5

Masukan Data Ke-1 = 34
Masukan Data Ke-2 = 12
Masukan Data Ke-3 = 67
Masukan Data Ke-4 = 54
Masukan Data Ke-5 = 31

Proses Selection Sort
[12] [34] [67] [54] [31]
[12] [31] [67] [54] [34]
[12] [31] [34] [54] [67]
[12] [31] [34] [54] [67]
[12] [31] [34] [54] [67]
```

c) Penjelasan

Selection sort : algoritma pengurutan sederhana yang bekerja dengan mencari elemen terkecil pada array dan memindahkan elemen tersebut ke posisi awal. Setelah itu, algoritma ini akan mencari elemen terkecil berikutnya dan memindahkan elemen tersebut ke posisi kedua, dan seterusnya. Proses ini diulangi hingga seluruh array terurut.

1. Mulai dari indeks pertama, cari elemen terkecil dalam array.
2. Tukar elemen terkecil dengan elemen di indeks pertama.
3. Mulai dari indeks kedua, cari elemen terkecil dalam array kecuali elemen pertama.
4. Tukar elemen terkecil dengan elemen di indeks kedua.
5. Ulangi proses di atas hingga seluruh array terurut.

5. Percobaan 5

a) Tampilan coding

```cpp
#include <iostream>
#include <conio.h>
using namespace std;

int data1[10], data2[10];
int n;

int tukar(int a, int b) {
    int t;
    t = data1[b];
    data1[b] = data1[a];
    data1[a] = t;
    return 0;
}

int input() {
    cout << "Masukan Jumlah Data = ";
    cin >> n;

    cout << endl;

    for (int i = 0; i < n; i++) {
        cout << "Masukan data ke -" << i + 1 << " = ";
        cin >> data1[i];

        data2[i] = data1[i];
    }
    cout << endl;
    return 0;
}

int tampil() {
    for (int i = 0; i < n; i++) {
        cout << "[" << data1[i] << "] ";
    }
    cout << endl;
    return 0;
}

int buuble_sort(int data1[], int n) {
    for (int i = 1; i < n; i++) {
        for (int j = n - 1; j >= i; j--) {
            if (data1[j] < data1[j - 1]) {
                tukar(j, j - 1);
            }
        }
        tampil();
    }
    cout << endl;
    return 0;
}
```

```
52
53  int selection_sort(int datal[], int n) {
54      int i, j, posisi, tukar;
55      for (i = 0; i < (n - 1); i++) {
56          posisi = i;
57          for (j = i + 1; j < n; j++) {
58              if (datal[posisi] > datal[j]) {
59                  posisi = j;
60              }
61          }
62          if (posisi != i) {
63              tukar = datal[i];
64              datal[i] = datal[posisi];
65              datal[posisi] = tukar;
66          }
67
68      }
69      return 0;
70  }
71
72  int insertionSort(int data[], int n)
73  {
74      int temp, j;
75      for (int i = 1; i < n; i++) {
76          temp = data[i];
77          j = i - 1;
78          while (data[j] > temp && j >= 0) {
79              data[j + 1] = data[j];
80              j--;
81          }
82          data[j + 1] = temp;
83          tampil();
84      }
85      cout << endl;
86  }
87
88  int main() {
89      //nim nama
90      cout << "Nama : Wijayanto Agung Wibowo" << endl;
91      cout << "NIM : 22.11.4552" << endl << endl;
92
93      cout << "ALGORITMA INSERTION SORT" << endl;
94      cout << "------------------------" << endl;
95      input();
96      cout << "Proses insertion sort" << endl;
97      tampil();
98      insertionSort();
99      return 0;
100 }
```

b) Hasil Runing

```
Nama : Wijayanto Agung Wibowo
NIM : 22.11.4552

ALGORITMA INSERTION SORT
----------------------
Masukan Jumlah Data = 5

Masukan Data Ke-1 = 34
Masukan Data Ke-2 = 23
Masukan Data Ke-3 = 12
Masukan Data Ke-4 = 34
Masukan Data Ke-5 = 5

Proses Insertion Sort
[23] [34] [12] [34] [5]
[12] [23] [34] [34] [5]
[12] [23] [34] [34] [5]
[5] [12] [23] [34] [34]

[5] [12] [23] [34] [34]
```

c) Penjelasan

Algoritma sorting sederhana yang bekerja dengan membandingkan satu per satu elemen dari sebuah array. Proses pengurutan dimulai dari elemen kedua, kemudian elemen-elemen selanjutnya dimasukkan ke dalam urutan yang tepat dalam sub-array yang diurutkan sebelumnya.

Langkah-langkah insertion sort:

1. Pertama, pertimbangkan elemen kedua dari array. Bandingkan elemen kedua dengan elemen pertama. Jika elemen kedua lebih kecil dari elemen pertama, maka tukar posisi keduanya.
2. Kemudian, pertimbangkan elemen ketiga. Bandingkan elemen ketiga dengan elemen kedua, dan jika elemen ketiga lebih kecil dari elemen kedua, tukar posisi keduanya. Lalu, bandingkan elemen kedua dengan elemen pertama, dan jika elemen kedua lebih kecil dari elemen pertama, tukar posisi keduanya.
3. Lanjutkan langkah kedua untuk elemen keempat, kelima, dan seterusnya, hingga seluruh elemen pada array terurut secara bertahap.

Dalam setiap iterasi, elemen saat ini dibandingkan dengan elemen-elemen pada sub-array yang telah diurutkan sebelumnya. Jika elemen saat ini lebih kecil dari elemen di posisi sebelumnya, maka elemen tersebut akan ditukar posisinya dengan elemen sebelumnya, dan begitu seterusnya sampai elemen saat ini ditempatkan pada posisi yang tepat.

6. Studi kasus 1

a) Tampilan coding

```
/**
 * program sequential search
 * dapat menampilkan semua index data yang ditemukan
 *
 */
#include <iostream>
using namespace std;
const int MAX_SIZE{ 20 };
int data1[MAX_SIZE]; // array data
int idx[MAX_SIZE]; // array untuk menyimpan index elemen yang ditemukan
int counter{ 0 }; // counter, menghitung ada berapa banyak data yang ditemukan


void search(int x, int n) {
    for (auto i = 0; i < n; ++i)
    {
        // jika x ditemukan pada data[i]
        if (x == data1[i])
        {
            // simpan index i ke array idx
            idx[counter++] = i;
        }
    }
}

int main(void) {
    //nim nama
    cout << "Nama : Wijayanto Agung Wibowo" << endl;
    cout << "NIM : 22.11.4552" << endl << endl;

    int n;
    cout << "Masukan jumlah data: ";
    cin >> n;
    for (auto i = 0; i < n; ++i) {
        cout << "data ke- " << i <<" = ";
        cin >> data1[i];
    }
    int x;
    cout << "Cari: ";
    cin >> x;
    search(x,n);
    // jika counter > 0, berarti ada data yang ditemukan
    if (counter > 0)
    {
        cout << "Data ditemukan pada index ke - : ";
        for (auto i = 0; i < counter; ++i)
        {
            cout << idx[i] << ", ";
        }
    }
    else
    {
        cout << "data tidak ditemukan\n";
    }
    return 0;
}

```

b) Hasil running

```
Nama : Wijayanto Agung Wibowo
NIM : 22.11.4552

Masukan jumlah data: 5
data ke- 0 = 3
data ke- 1 = 5
data ke- 2 = 5
data ke- 3 = 7
data ke- 4 = 9
Cari: 5
Data ditemukan pada index ke - : 1, 2,
D:¥kuliah¥SEMESTER 2¥Struktur Data -laptop¥CONSOLE¥Project1¥x64¥Debug¥Project1.exe (process 17496) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

```
Nama : Wijayanto Agung Wibowo
NIM : 22.11.4552

Masukan jumlah data: 5
data ke- 0 = 3
data ke- 1 = 5
data ke- 2 = 5
data ke- 3 = 7
data ke- 4 = 6
Cari: 9
data tidak ditemukan

D:¥kuliah¥SEMESTER 2¥Struktur Data -laptop¥CONSOLE¥Project1¥x64¥Debug¥Project1.exe (process 5384) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

c) Penjelasan

Program diatas adalah program untuk memasukan data dan mencari data secara dinamis. Menentukan maksimal data dengan variable n. setelah itu dimasukan kedalam for, untuk memasukan data secara dinamis berdasarkan inputan yang kita masukan. setelah memasukan nilai kedalam array, kita memasukan nilai data yang dicari. Jika data ditemukan, maka akan mengakibatkkan counter ++ dan apabila meng cout kan maka akan menampilkan data yang ada. Jika tidak ada data akan menampilkan hasil “Data tidak ditemukan.

7. Studi kasus 2

a) Tampilan coding

```cpp
#include <iostream>
#include <ctime>
#include <windows.h>
using namespace std;
const int ukuran{ 25 };
const int BUBBLE{ 0 };
const int SELECTION{ 1 };
const int INSERTION{ 2 };
int nsrc[ukuran];
int ndata[ukuran];
void init_data();
void load_data();
void view_data();
void bubsort();
void selsort();
void inssort();
void run_sort(int);
int main(void) {
    cout << "Perbandingan Metode Pengurutan\n";
    init_data();
    load_data();
    cout << "Data Acak:\n";
    view_data();
    cout << "\n\n";
    // Bubble Sort
    cout << "Bubble Sort\n";
    load_data();
    run_sort(BUBBLE);
    // Selection Sort
    cout << "Selection Sort\n";
    load_data();
    run_sort(SELECTION);
    // Insertion Sort
    load_data();
    cout << "Insertion Sort\n";
    run_sort(INSERTION);
    return 0;
}
void init_data(void)
{
    srand(static_cast<unsigned int>(time(NULL)));
    for (int i = 0; i < ukuran; ++i)
        nsrc[i] = rand() % 100;
}
void load_data(void)
{
    for (int i = 0; i < ukuran; ++i)
        ndata[i] = nsrc[i];
}
void view_data(void)
{
    for (int i = 0; i < ukuran; ++i)
        printf("%d ", ndata[i]);
    cout << endl;
}
```

```
56  void bubsort(void)
57  {
58      for (int i = 0; i < ukuran - 1; ++i)
59      {
60          for (int j = 0; j < ukuran - i - 1; ++j)
61          {
62              if (ndata[j] > ndata[j + 1])
63              {
64                  int tmp = ndata[j];
65                  ndata[j] = ndata[j + 1];
66                  ndata[j + 1] = tmp;
67                  Sleep(10);
68              }
69              Sleep(10);
70          }
71          Sleep(10);
72      }
73  }
74  void selsort(void)
75  {
76      for (int i = 0; i < ukuran; ++i)
77      {
78          int min = i;
79          for (int j = i; j < ukuran; ++j)
80          {
81              if (ndata[j] < ndata[min])
82              {
83                  min = j;
84                  Sleep(10);
85              }
86              Sleep(10);
87          }
88          int tmp = ndata[i];
89          ndata[i] = ndata[min];
90          ndata[min] = tmp;
91          Sleep(10);
92      }
93  }
94  void inssort(void)
95  {
96      for (int i = 1; i < ukuran; ++i)
97      {
98          int m = ndata[i];
99          auto s = i;
100         while (s >= 0 && m < ndata[s - 1])
101         {
102             ndata[s] = ndata[--s];
103             Sleep(10);
104         }
105         ndata[s] = m;
106         Sleep(10);
107     }
108 }
```

```
109   void run_sort(int m)
110   {
111       auto t1 = time(NULL);
112       switch (m)
113       {
114       case 0: bubsort(); break;
115       case 1: selsort(); break;
116       case 2:
117       default: inssort(); break;
118       }
119       auto t2 = time(NULL);
120       auto t = t2 - t1;
121       cout << "data terurut:\n";
122       view_data();
123       cout << "waktu: " << t << " milisecond\n";
124       cout << "----------------------------\n\n";
125   }
```

b) Hasil running

```
Perbandingan Metode Pengurutan
Data Acak:
81 15 43 15 28 81 42 58 29 52 19 38 7 74 32 23 67 54 6 73 68 94 8 8 79

Bubble Sort
data terurut:
6 7 8 8 15 15 19 23 28 29 32 38 42 43 52 54 58 67 68 73 74 79 81 81 94
waktu: 7 milisecond
--------------------------

Selection Sort
data terurut:
6 7 8 8 15 15 19 23 28 29 32 38 42 43 52 54 58 67 68 73 74 79 81 81 94
waktu: 6 milisecond
--------------------------

Insertion Sort
data terurut:
6 15 15 15 19 29 42 58 29 52 19 38 7 23 32 23 54 54 6 8 68 94 8 8 79
waktu: 1 milisecond
--------------------------


D:¥kuliah¥SEMESTER 2¥Struktur Data -laptop¥CONSOLE¥Project1¥x64¥Debug¥Project1.exe (process 13644) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

c) Penjelasan

Program diatas adalah untuk membandingkan algoritma sorting: buuble sort, selection sort, dan insertion sort. Pada awal program, terdapat deklarasi variabel konstan **ukuran** yang bernilai 25 dan tiga konstanta **BUBBLE**, **SELECTION**, dan **INSERTION** yang masing-masing bernilai 0, 1, dan 2. Kemudian, terdapat deklarasi array integer **nsrc** dan **ndata** yang masing-masing memiliki ukuran **ukuran**. **nsrc** berfungsi sebagai array sumber yang berisi data integer acak, sementara **ndata** berfungsi sebagai array data yang akan diurutkan.

Fungsi **init_data()** digunakan untuk menginisialisasi array **nsrc** dengan data integer acak menggunakan fungsi **rand()** dan **time()**.

Fungsi **load_data()** digunakan untuk memuat data dari array **nsrc** ke array **ndata**.

Fungsi **view_data()** digunakan untuk menampilkan data dalam array **ndata**.

Fungsi **bubsort()** digunakan untuk mengurutkan data dalam array **ndata** menggunakan metode Bubble Sort.

Fungsi **selsort()** digunakan untuk mengurutkan data dalam array **ndata** menggunakan metode Selection Sort.

Fungsi **inssort()** digunakan untuk mengurutkan data dalam array **ndata** menggunakan metode Insertion Sort.

Fungsi **run_sort()** menerima parameter integer **m** yang menunjukkan metode pengurutan mana yang akan dijalankan. Fungsi ini akan mencatat waktu sebelum dan sesudah pengurutan menggunakan fungsi **time()** dan menjalankan fungsi pengurutan yang sesuai dengan nilai **m**. Setelah pengurutan selesai, fungsi **run_sort()** akan menampilkan data terurut dalam array **ndata** dan waktu yang diperlukan untuk pengurutan.

Pada bagian utama program, terdapat panggilan fungsi **init_data()** untuk menginisialisasi array **nsrc**, dan panggilan **load_data()** untuk memuat data dari **nsrc** ke **ndata**. Kemudian, program akan menampilkan data yang akan diurutkan menggunakan fungsi **view_data()**. Program akan menjalankan ketiga metode pengurutan menggunakan fungsi **run_sort()** dengan nilai parameter **m** yang berbeda-beda. Setelah pengurutan selesai, program akan menampilkan data terurut dan waktu yang dibutuhkan untuk masing-masing pengurutan

C. Kesimpulan
   Setelah melakukan percobaan pada Latihan 1 dst saya dapat memahami bahwa ada banyak cara sort method dan pencarian data.
   Tergantung bagaimana pengimplementasian nya, Semua memiliki kelebihan dan kekurangannya masing-masing.