

**LAPORAN HASIL PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN II**



**NAMA : ELNATAN KENINGATKO
NIM : 193020503038
KELAS : A
MODUL : I (DASAR PEMROGRAMAN
BERORIENTASI OBJEK)**

**JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PALANGKA RAYA
2020**

**LAPORAN HASIL PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN II**



NAMA : ELNATAN KENINGATKO
NIM : 193020503038
KELAS : A
MODUL : I (DASAR PEMROGRAMAN
BERORIENTASI OBJEK)

Komposisi	MAX	Nilai
BAB I Tujuan dan Landasan Teori	10	9
BAB II Pembahasan	60	53
BAB III Kesimpulan	20	15
Daftar Pustaka	5	5
Lampiran	5	5
Jumlah	100	87

Penilai
Asisten Praktikum


Diana

BAB I

TUJUAN DAN LANDASAN TEORI

1.1. Tujuan

Setelah menyelesaikan modul ini, mahasiswa diharapkan mampu :

- a. Memahami dasar-dasar pemrograman berorientasi objek
- b. Memahami enkapsulasi
- c. Membuat kelas dan objek

1.2. Landasan Teori

1.2.1. Pengertian Pemrograman Berorientasi Objek

Pemrograman Berorientasi Objek atau Object Oriented Programming (OOP) adalah sebuah tata cara pembuatan program (programming paradigm) dengan menggunakan konsep “objek” yang memiliki data (atribut yang menjelaskan tentang objek) dan prosedur (function) yang dikenal dengan method.

Dalam pengertian sederhananya, OOP adalah konsep pembuatan program dengan memecah permasalahan program dengan menggunakan objek. Objek dapat diumpamakan dengan ‘fungsi khusus’ yang bisa berdiri sendiri. Untuk membuat sebuah aplikasi, berbagai objek akan saling bertukar data untuk mencapai hasil akhir. Berbeda dengan konsep fungsi atau ‘function’ di dalam pemrograman, sebuah objek bisa memiliki data dan function tersendiri. Setiap objek ditujukan untuk mengerjakan sebuah tugas, dan menghasilkan nilai akhir untuk selanjutnya dapat ditampilkan atau digunakan oleh objek lain.

1.2.2. Manfaat Pemrograman Berorientasi Objek

PBO diciptakan karena masih dirasakan adanya keterbatasan pada bahasa pemrograman tradisional yang dikenal dengan istilah procedural language seperti C, Pascal dan yang sejenisnya.

Padahal, pemrograman prosedural sendiri pada awalnya merupakan perbaikan atas bahasa pemrograman sekuensial (sequential programming language), seperti BASIC ataupun Fortran. Konsep procedural programming language yaitu bahwa semua masalah dibagi ke dalam fungsi atau prosedur. Selain itu procedural programming, fungsi dan data terpisah dan tidak menjadi satu kesatuan.

Contoh kelemahan dari procedural programming adalah bila sebuah perusahaan membuat program General Ledger (Buku kas besar) menggunakan variabel yang bisa diakses oleh fungsi-fungsi lainnya, maka harus dideklarasikan variabel tersebut sebagai variabel global yang letaknya di luar semua fungsi sehingga bisa diakses oleh semua fungsi. Jika suatu saat perusahaan tadi memperkerjakan programmer baru yang belum mengetahui seluk-beluk program general ledger tadi, maka sangat dimungkinkan terjadinya kerusakan data. Anda juga akan menemukan kesulitan bila Anda ingin membuat tipe data baru dengan bahasa pemrograman tradisional.

Konsep PBO adalah bahwa semua pemecahan masalah dibagi ke dalam kelas (class). Dalam PBO data dan fungsi-fungsi yang akan mengoperasikan data digabungkan menjadi satu kesatuan yang bisa disebut sebagai kelas.

Fungsi kelas terletak pada fungsi anggota (member function) dalam Java. Jika Anda ingin membaca data dalam objek maka Anda harus memanggil fungsi anggota (member function) dalam objek. hanya dengan cara ini Anda bisa mengakses data. Jadi Anda tidak bisa mengakses data secara langsung seperti dalam konsep pemrograman tradisional. Pengaksesan data seperti pada PBO ini dikenal sebagai pengkapsulan data. Pengkapsulan data (data encapsulation) dan penyembunyian data merupakan kunci pengertian pemrograman berorientasi objek.

Jika Anda ingin mengubah data dalam objek tentunya Anda harus tahu fungsi-fungsi apa saja yang berinteraksi dengan objek itu,

jadi tidak ada fungsi-fungsi lain yang bisa mengakses data sehingga penulisan dan penelusuran kesalahan program (debugging) akan dapat dilakukan dengan mudah.

Untuk mempermudah pengertian mengenai objek, dimisalkan objek sebagai departemen-departemen dalam perusahaan, seperti pemasaran, keuangan, produksi, personalia, pengadaan maupun warehouse. Setiap departemen memiliki fungsi, tugas dan tanggung jawab yang berbeda. Jika menginginkan data laporan keuangan maka yang harus diminta adalah departemen keuangan, bukan yang lainnya.

Secara umum beberapa keuntungan yang tampak pada pemrograman berorientasi objek (OOP) adalah :

- a. Objek-objeknya dapat digunakan ulang (reusable) untuk program-program lain.
- b. Programnya lebih terstruktur dan lebih mudah untuk dikembangkan
- c. Bersifat alami
- d. Reusabilitas
- e. Pembangunan program lebih cepat
- f. Fleksibilitas lebih tinggi
- g. Ekstensibilitas
- h. Less Maintenance

1.2.3. Pengertian Class

Class merupakan blueprint (cetak biru) untuk menciptakan suatu instance dari objek dimana terdiri dari sekumpulan objek dengan kemiripan data / properties / attributes, fungsi / behavior / method dan relasi ke objek lain. Pemrograman C++ memungkinkan pembuatan class lebih dari 1. Ketika data dan fungsi yang terkait disimpan di dalam sebuah class mampu membantu memvisualisasikan permasalahan yang kompleks dengan efisien dan efektif. Contoh : *class* Mahasiswa, *class* Dosen, dll.

Data dan fungsi yang berada di dalam sebuah class disebut sebagai anggota dari suatu class. Data pada suatu class digunakan untuk memegang informasi yang ada pada class tersebut, sedangkan fungsi digunakan sebagai behavior dari class tersebut. Untuk pembuatan sebuah class, dimulai dengan kata kunci class dan diikuti dengan nama kelasnya, dibuka dengan {, isidari class, ditutup dengan };. Berikut adalah sintaks pembuatan class:

```
class class_name {  
    data members;  
    methods;  
};
```

Contoh Program :

```
class box {  
    public:  
    int length; //Panjang Box  
    int width; //Lebar Box  
    int height; //Tinggi Box  
};
```

Seperti yang telah disebutkan di atas, pendefinisian suatu kelas dimulai dengan kata kunci class dan diikuti dengan nama kelas Box dalam kasus ini. Isi dari suatu kelas ditandai dengan { dan diakhiri dengan } diikuti;. Kata kunci public pada contoh di atas menentukan cara pengaksesan anggota kelas. Anggota kelas public dapat diakses di kelas manapun.

1.2.4. Member Class

Seperti yang telah disinggung sebelumnya bahwa data dan fungsi di dalam suatu class disebut dengan anggota suatu class. Perhatikan contoh berikut ini :

```
class box {  
    public:
```

```
int length; //Panjang Box
int breadth; //Luas Box
int height; //Tinggi Box

void print() {
    cout<<"Printing Box Object"<<endl;
}
};
```

Berdasarkan contoh di atas, `length`, `breadth` dan `height` merupakan data member dari class `box`; sedangkan `print` merupakan function member (member fungsi) dari class `box`.

1.2.5. Pengenalan Objek

Jika class menyediakan blueprint untuk membuat objek, maka, secara dasarnya, objek dibentuk dari suatu class. Pada intinya, objek adalah suatu kumpulan yang memiliki atribut dan metode yang sama (instance dari class). Dalam konteks variabel, suatu class dapat dianggap sebagai tipe data, dan objek sebagai variabelnya. Contoh: Dari class `Flowers` dapat dihasilkan objek `Rose`, `Orchid`, `SunFlower`, dsb. Sintaks untuk membuat objek adalah :

```
class_name variable_name;
```

Contohnya :

```
box obj1,obj2;
```

Berdasarkan contoh di atas, terdapat dua buah objek yang berasal dari kelas `Box`, yaitu `obj1` dan `obj2`.

1.2.6. Mengakses Data Member dan Function Member

Data member dan *member function* (anggota data dan fungsi) dapat diakses dengan menggunakan operator `(.)`. Secara umum,

sintaks untuk mengakses *data member* ataupun *function member* adalah :

```
object_name.data_member;
```

Contoh program dari keseluruhan kode yang telah dibahas :

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

class box {
    public:
    int length; //Panjang Box
    int width; //Lebar Box
    int height; //Tinggi Box

    void print() {
        cout<<"Printing Box Object"<<endl;
    }
};

int main() {
    box obj1,obj2; //Deklarasi 2 objek obj1 dan obj2 kelas box
    int volume=0; //Menyimpan volume dari box

    //Spesifikasi box obj1
    obj1.height=4;
    obj1.length=6;
    obj1.width=3;

    //Spesifikasi box obj2
    obj2.height=10;
    obj2.length=12;
    obj2.width=12;

    //volume box obj1
    volume=obj1.height*obj1.length*obj1.width;
    cout<<"Volume of box1 : <<"volume<<endl;

    //volume box obj2
    volume=obj2.height*obj2.length*obj2.width;
```



```
cout<<"Volume of box2 : <<"volume<<endl;  
}
```

Outputnya adalah :

```
Volume of box1 : 72  
Volume of box2 : 1440
```

BAB II

PEMBAHASAN

2.1. Program Pertama

```
#include <iostream>
#include <conio.h>

using namespace std;

class buah
{
public:
    buah(string, string, string);
    void cetakbuah(string);
    void cetakwarna(string);
    void cetakrasa(string);
    void info();
private:
    string namabuah;
    string warna;
    string rasa;
};

buah::buah(string n, string w, string r)
{
    namabuah=n;
    warna=w;
    rasa=r;
}

void buah::cetakbuah(string n)
{
    namabuah=n;
}

void buah::cetakwarna(string w)
{
    warna=w;
}
```

```

void buah::cetakrasa(string r)
{
    rasa=r;
}

void buah::info()
{
    cout<<"Buah yang dipilih adalah "<<namabuah<<endl;
    cout<<"Warna buah "<<namabuah<<" adalah "<<warna<<endl;
    cout<<"Rasa buah "<<namabuah<<" adalah
    "<<namabuah<<endl<<endl;
}
int main()
{
    buah buah1("semangka", "hijau", "manis"),
    buah2("melon", "hijau", "manis"),
    buah3("mangga", "hijau", "masam");

    buah1.info();
    buah2.info();
    buah3.info();
}

```

Penjelasan

```

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

```

Code di atas merupakan langkah awal dari sebuah coding dalam bahasa pemrograman C++. *#include* merupakan sebuah prosesor pengarah yang mengatakan kepada kompiler untuk meletakkan kode dari header file *iostream* dan *conio.h* kedalam program. Header “*iostream*” memiliki arti Input Output Stream yang digunakan untuk memanggil fungsi pada library berupa fungsi input yaitu *cin*, fungsi output yaitu *cout* dan fungsi untuk membuat garis baru yaitu *endl*. Header “*conio.h*” digunakan untuk menampilkan hasil antarmuka kepada user, seperti *getch()* yang berfungsi untuk menahan output suatu

program dan akan kembali mengeksekusi setelah user meng-input dan kemudian menekan enter atau tombol lainnya, *getche()* yang fungsinya sama dengan *getch()* akan tetapi ketika melakukan input-an, maka akan tampil dalam window dan *clrscr()* yang merupakan singkatan dari clear screen yang digunakan untuk membersihkan layar windows.

Using namespace std; digunakan untuk memberitahukan kepada kompiler bahwa kita akan menggunakan semua fungsi, class atau file yang terdapat pada memori *namespace std*. Sehingga tidak perlu menambahkan *std::* di depan fungsi-fungsi untuk setiap penggunaan fungsi seperti fungsi *cout*, *cin* dan sebagainya.

```
class buah
{
public:
    buah(string, string, string);
    void cetakbuah(string);
    void cetakwarna(string);
    void cetakrasa(string);
    void info();
private:
    string namabuah;
    string warna;
    string rasa;
};
```

Code di atas merupakan kelas dengan objek "buah". Kelas tersebut memiliki akses *public* dengan methods berupa *void cetakbuah(string);*, *void cetakwarna(string);*, *void cetakrasa(string);*, *void info();* dan satu constructor yaitu *buah(string, string, string);* yang mana memiliki nama yang sama dengan nama kelas. Fungsi "(string)" yaitu sebagai tanda bahwa methods tersebut memiliki tipe data string (teks). Kemudian akses *private* dengan methods berupa *string namabuah;*, *string warna;*, dan *string rasa;*.

```

buah::buah(string n, string w, string r)
{
    namabuah=n;
    warna=w;
    rasa=r;
}
void buah::cetakbuah(string n)
{
    namabuah=n;
}

void buah::cetakwarna(string w)
{
    warna=w;
}

void buah::cetakrasa(string r)
{
    rasa=r;
}

```

Code “buah::buah(string n, string w, string r)” merupakan *member functions* dari *constructor* yang digunakan untuk menginisialisasi methods, sehingga methods *namabuah* menjadi *n*, *warna* menjadi *w* dan *rasa* menjadi *r*. Void merupakan fungsi yang tidak memiliki nilai balikan atau digunakan untuk pemanggilan fungsi di badan utama program. Contohnya code “void buah::cetakbuah(string n)”, code tersebut memiliki *member functions* “cetakbuah(string n)” yang diakses ke kelas *buah*, dihubungkan tanda “::” (*Binary Scope Resolution Operator*) dengan kelas *buah* yang berfungsi untuk “mengikat” nama fungsi ke nama kelas dan mengidentifikasi fungsi dari kelas *buah*. Kemudian code “namabuah=n;” merupakan inisialisasi yang digunakan *member functions* sebagai parameter penyimpanan data pada saat user meng-input data.

```

void buah::info()
{

```

```
cout<<"Buah yang dipilih adalah "<<namabuah<<endl;
cout<<"Warna buah "<<namabuah<<" adalah "<<warna<<endl;
cout<<"Rasa buah "<<namabuah<<" adalah
"<<namabuah<<endl<<endl;
}
```

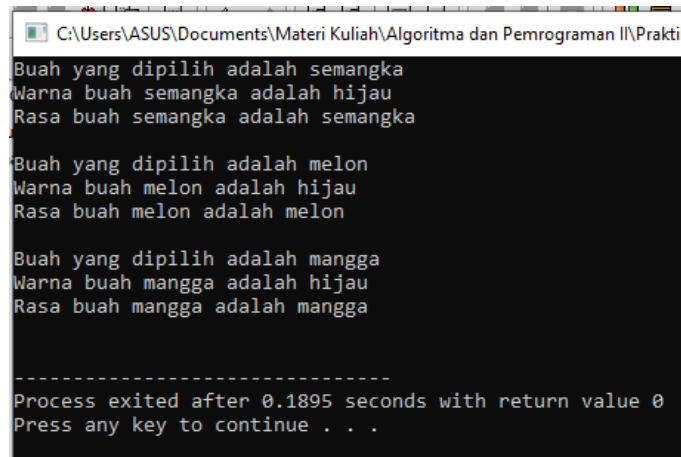
Code di atas juga merupakan *member functions*. Bedanya, *member functions* tersebut merupakan pemanggilan fungsi yang berisikan output untuk user meng-input data. Code “cout<<”Buah yang dipilih adalah”<<namabuah<<endl;” menampilkan output “Buah yang dipilih adalah (data yang diinput user)”. “<<namabuah” merupakan script yang digunakan untuk menampilkan output yang mana merupakan data hasil dari input-an user, sedangkan “<<endl;” merupakan script yang digunakan untuk menandakan bahwa baris tersebut telah selesai dieksekusi dan akan berpindah ke baris selanjutnya.

```
int main()
{
    buah buah1("semangka", "hijau", "manis"),
    buah2("melon", "hijau", "manis"),
    buah3("mangga", "hijau", "masam");

    buah1.info();
    buah2.info();
    buah3.info();
}
```

Code di atas merupakan badan utama program dengan menggunakan fungsi “int main()” sebagai perintah untuk memulai eksekusi program. Dengan *buah* sebagai kelas dan *buah1* yang berisikan data “semangka, hijau, manis” sebagai objek. Data tersebut dimasukan ke dalam fungsi yang telah dibuat sesuai dengan urutannya. Code “buah1.info();” merupakan pemanggilan fungsi yang telah dibuat dan menampilkan output dengan

menggunakan data yang telah di-input oleh user secara berurutan. Berikut hasil output dari program pertama :



```
C:\Users\ASUS\Documents\Materi Kuliah\Algoritma dan Pemrograman II\Praktik
Buah yang dipilih adalah semangka
Warna buah semangka adalah hijau
Rasa buah semangka adalah semangka

Buah yang dipilih adalah melon
Warna buah melon adalah hijau
Rasa buah melon adalah melon

Buah yang dipilih adalah mangga
Warna buah mangga adalah hijau
Rasa buah mangga adalah mangga

-----
Process exited after 0.1895 seconds with return value 0
Press any key to continue . . .
```

Gambar 2.1. Output Program Pertama

2.2. Program Kedua

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

class buah
{
public:
    buah (string, string);
    void cetaknama(string);
    void cetakbuah(string);
    void info();
private:
    string namaorang;
    string buahorang;
};

buah::buah(string n, string b)
{
    namaorang=n;
    buahorang=b;
}
```

```

void buah::cetaknama(string n)
{
    namaorang=n;
}

void buah::cetakbuah(string b)
{
    buahorang=b;
}

void buah::info()
{
    cout<<"Nama anda adalah "<<namaorang<<endl;
    cout<<"Buah yang anda pilih adalah "<<buahorang<<endl;
}

int main()
{
    string inputnama, inputbuah;
    cout<<"Masukan nama anda          : "; getline(cin, inputnama);
    cout<<"Pilih buah yang anda inginkan : "; getline(cin, inputbuah);
    system("cls");
    buah hasil=buah(inputnama, inputbuah);
    hasil.info();

    getch();
    return 0;
}

```

Penjelasan

Program tersebut memiliki struktur yang sama. Tetapi memiliki perbedaan di bagian badan utama program. Jika di program pertama data input user dimasukan langsung di dalam source code, maka di program kedua ini user bisa meng-input data sesuai keinginannya di output program. Berikut penjelasan perbedaannya :


```

int main()
{
    string inputnama, inputbuah;
    cout<<"Masukan nama anda      : "; getline(cin, inputnama);
    cout<<"Pilih buah yang anda inginkan : "; getline(cin, inputbuah);
    system("cls");
    buah hasil=buah(inputnama, inputbuah);
    hasil.info();

    getch();
    return 0;
}

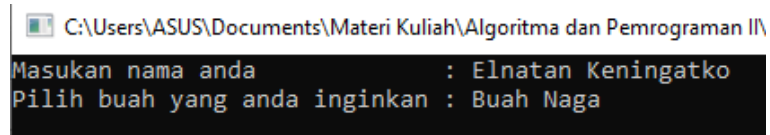
```

Perbedaan source code badan utama program pertama dan program kedua yaitu, terdapat variable *inputnama* dan *input buah* yang bertipe data string. Kegunaannya yaitu untuk menampung data yang akan di-input oleh user pada saat running program. Code “cout<<"Masukan nama anda : "; getline(cin, inputnama);” mengharuskan user untuk memasukan teks yang akan di tampung di variable *inputnama*. Kegunaan *getline* yaitu agar user bisa meng-input teks lebih dari satu kata, jika tidak menggunakan *getline* maka hanya satu kata saja yang dapat dimasukkan ke dalam variable *inputnama* dan kata selanjutnya akan otomatis masuk ke variable selanjutnya.

System("cls"); digunakan untuk membersihkan layar, artinya ketika user selesai meng-input data maka pada saat menekan tombol enter layar akan bersih dan kemudian mengeluarkan output dengan source code yang ada setelah script “system(“cls”);”.

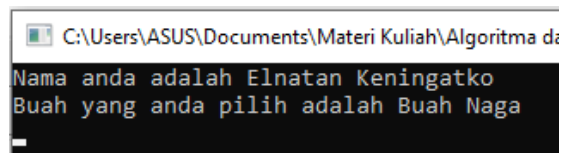
Code “buah hasil=buah(inputnama, inputbuah);” merupakan kelas (buah) dan objek (hasil) yang kemudian dihubungkan tanda “=” ke code “buah(inputnama, inputbuah)” yang memiliki kegunaan untuk menandakan bahwa methods yang terdapat di kelas *buah* diisi dengan variable *inputnama* dan *inputbuah* yang memiliki data hasil input dari user, data tersebut dimasukan ke dalam methods kelas sesuai urutannya. Kemudian untuk menampilkan hasil data yang telah di-input user menggunakan fungsi *info()*; yang dihubungkan dengan kelas *buah* menggunakan tanda “.” agar functions

dapat diakses dari dalam kelas (*Member Access Operators*). Berikut ini hasil output program kedua :



```
C:\Users\ASUS\Documents\Materi Kuliah\Algoritma dan Pemrograman II\
Masukan nama anda           : Elnatan Keningatko
Pilih buah yang anda inginkan : Buah Naga
```

Gambar 2.2. Output Program Kedua (1)



```
C:\Users\ASUS\Documents\Materi Kuliah\Algoritma dan Pemrograman II\
Nama anda adalah Elnatan Keningatko
Buah yang anda pilih adalah Buah Naga
_
```

Gambar 2.3. Output Program Kedua (2)

BAB III

KESIMPULAN

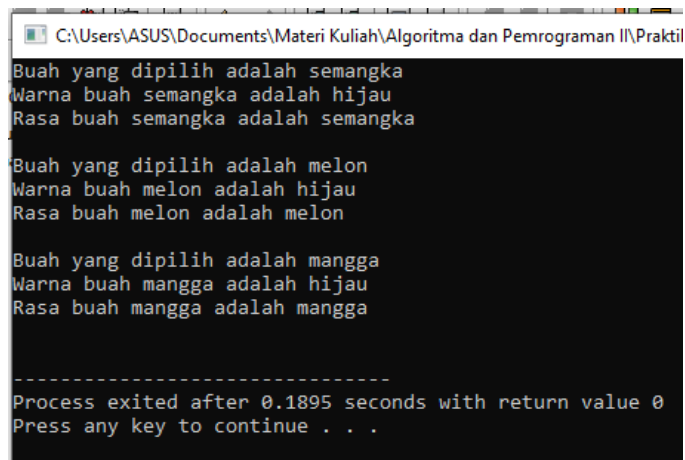
Dapat di tarik kesimpulan sebagai berikut, Pemrograman Berorientasi Objek (PBO) merupakan pemrograman yang berorientasikan terhadap objek, dimana semua data dan fungsi yang dibungkus ke dalam kelas-kelas atau objek-objek. dan enkapsulasi adalah Pembungkusan variabel dan method dalam sebuah obyek yang terlindungi serta menyediakan interface untuk mengakses variabel tersebut. Variabel dan method yang dimiliki oleh suatu objek, bisa ditentukan hak aksesnya.

Objek berfungsi untuk membungkus data dan fungsi bersama menjadi satu unit dalam sebuah program komputer. Objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek.

DAFTAR PUSTAKA

- Aliffina, Sylva. 2016. *Konsep Dasar PBO (Pemrograman Berorientasi Objek)*, diakses dari <https://sylvaputr.wordpress.com/2016/09/01/pbo-pemograman-berorientasi-objek/>, pada 07 April 2020.
- Harefa, Jeklin. 2016. *Class and Object : Object Oriented Programming Using C++*, diakses dari <https://socs.binus.ac.id/2016/12/13/class-and-object-object-oriented-programming-using-c/>, pada 07 April 2020.
- Tim Penyusun : Dosen Teknik Informatika. 2020. *Modul Praktikum Algoritma dan Pemrograman II*. Palangka Raya : Jurusan Teknik Informatika.

LAMPIRAN



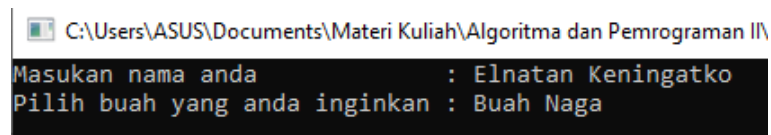
```
C:\Users\ASUS\Documents\Materi Kuliah\Algoritma dan Pemrograman II\Praktik
Buah yang dipilih adalah semangka
Warna buah semangka adalah hijau
Rasa buah semangka adalah semangka

Buah yang dipilih adalah melon
Warna buah melon adalah hijau
Rasa buah melon adalah melon

Buah yang dipilih adalah mangga
Warna buah mangga adalah hijau
Rasa buah mangga adalah mangga

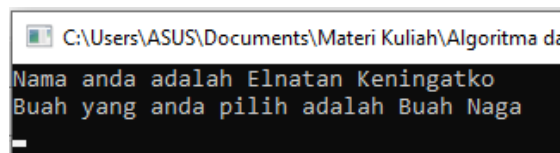
-----
Process exited after 0.1895 seconds with return value 0
Press any key to continue . . .
```

Gambar 2.1. Output Program Pertama



```
C:\Users\ASUS\Documents\Materi Kuliah\Algoritma dan Pemrograman II\
Masukan nama anda : Elnatan Keningatko
Pilih buah yang anda inginkan : Buah Naga
```

Gambar 2.2. Output Program Kedua (1)



```
C:\Users\ASUS\Documents\Materi Kuliah\Algoritma dan Pemrograman II\
Nama anda adalah Elnatan Keningatko
Buah yang anda pilih adalah Buah Naga
```

Gambar 2.3. Output Program Kedua (2)