# LAPORAN HASIL PRAKTIKUM
# ALGORITMA DAN PEMROGRAMAN II


**NAMA** : **EVAN ALPHARIO IMANUEL**
**NIM** : **193030503059**
**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 : EVAN ALPHARIO IMANUEL

NIM : 193030503059

Kelas : A

Modul : I ( PBO )

| Komposisi | MAX | Nilai |
|---|---|---|
| BAB I Tujuan dan Landasan Teori | 10 | 7 |
| BAB II Pembahasan | 60 | 50 |
| BAB III Kesimpulan | 20 | 10 |
| Daftar Pustaka | 5 | 3 |
| Lampiran | 5 | 3 |
| Jumlah | 100 | 73 |

Penilai

Asisten Praktikum

Diana

# BAB I
# TUJUAN DAN LANDASAN TEORI

## I. TUJUAN

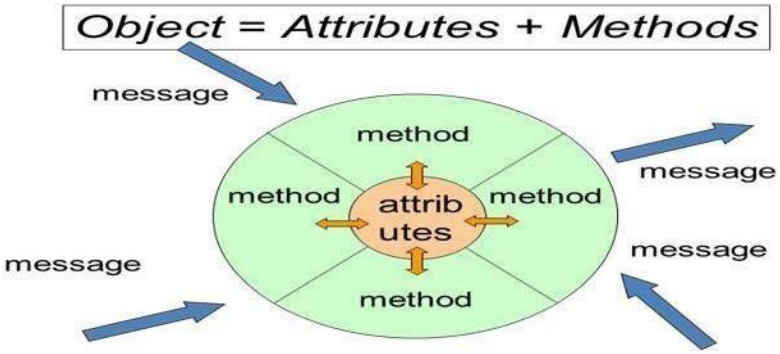
Setelah menyelesaikan modul ini, mahasiswa diharapkan mampu:

- Memahami dasar-dasar pemrograman berorientasi obyek
- Memahami enkapsulasi
- membuat kelas dan objek

## II. DASAR TEORI

Perbedaan pemrograman tradisional dan berorientasi objek adalah pada cara menelesaikan suatu permasalahan. Pada pemrograman tradisional dalam memecahkan suatu masalah, masalah akan dibagi menjadi fungsi-fungsi yang lebih kecil, sedangkan pada pemrograman berorientasi objek (PBO) setiap masalah diselesaikan dengan cara dibagi ke dalam objek–objek.

Pada PBO dilakukan pembungkusan data (attributes) dan fungsi (behavior) ke paket yang disebut kelas. *Attributes* merupakan data yang menggambarkan status internal sebuah objek dan biasanya merupakan **"***member variables***"** pada C++, tidak dapat diakses dari luar (enkapsulasi), dan juga sebagai **"**state**"**. *Methods* merupakan fungsi yang mengakses status internal sebuah objek dan biasanya merupakan **"***member functions***"** pada C++, dapat diakses dari luar, memanipulasi atribut, dan disebut juga **"**behavior**"**. Berikut ini merupakan gambaran mengenai objek.




Kelas (Class) terdiri dari model objek yang memiliki atribut (data members) dan *Behaviors* (*member functions*), dan *Member functions* yaitu *Methods* yang dipanggil sebagai response terhadap pesan. Kelas didefinisikan dengan *keyword* **class.**

Mode Akses akses yang ada pada kelas ada tiga yaitu ***private*** yang merupakan *default* mode akses dan dapat diakses oleh *member functions*, ***public*** yang dapat diakses oleh setiap Accessible fungsi dalam program, dan ***protected*** yang biasanya digunakan untuk pewarisan .

Fungsi *Constructor* merupakan *member function* khusus yang menginisialisasi data members dan memiliki nama yang sama dengan nama kelas. Fungsi *Constructor d*ipanggil saat membuat objek dari kelas dan tidak memiliki tipe balikan.

*Member functions* yang didefinisikan di luar kelas dilakukan dengan menggunakan *binary scope resolution operator* (**::**) yang berfungsi untuk **"**mengikat**"** nama fungsi ke nama kelas dan mengindentifikasi fungsi dari suatu kelas tertentu.

Berikut ini merupakan format dari *member functions*.

**NilaiBalikan *NamaKelas::NamaFungsi( ){***

**…**

***}***

*Member functions* yang didefinisikan di dalam kelas tidak membutuhkan scope resolution operator dan nama kelas.

**Konsep dasar**

- Kelas : kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebagai contoh 'class of dog' adalah suatu unit yang terdiri atas definisi-definisi data dan fungsi-fungsi yang menunjuk pada berbagai macam perilaku/turunan dari anjing. Sebuah class adalah dasar dari modularitas dan struktur dalam pemrograman berorientasi object. *Sebuah class secara tipikal sebaiknya dapat dikenali oleh seorang non-programmer sekalipun terkait dengan domain permasalahan yang ada,* dan kode yang terdapat dalam sebuah class sebaiknya (relatif) bersifat mandiri dan independen (sebagaimana kode tersebut digunakan jika tidak menggunakan OOP). Dengan modularitas, struktur dari sebuah program akan terkait dengan aspek-aspek dalam masalah yang akan diselesaikan melalui program tersebut. Cara seperti ini akan menyederhanakan pemetaan dari masalah ke sebuah program ataupun sebaliknya.
- Objek : membungkus data dan fungsi bersama menjadi suatu unit dalam sebuah program komputer; objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek.
- Abstraksi : Kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari "pelaku" abstrak yang dapat melakukan kerja, laporan dan perubahan keadaannya, dan berkomunikasi dengan objek lainnya dalam sistem, tanpa mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak, dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan.
- Enkapsulasi : Memastikan pengguna sebuah objek tidak dapat mengganti keadaan dalam dari sebuah objek dengan cara yang tidak layak; hanya metode dalam objek tersebut yang diberi izin untuk mengakses keadaannya. Setiap objek mengakses interface yang menyebutkan bagaimana objek lainnya dapat

berinteraksi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut.

- Polimorfisme melalui pengiriman pesan. Tidak bergantung kepada pemanggilan subrutin, bahasa orientasi objek dapat mengirim pesan; metode tertentu yang berhubungan dengan sebuah pengiriman pesan tergantung kepada objek tertentu di mana pesa tersebut dikirim. Contohnya, bila sebuah burung menerima pesan "gerak cepat", dia akan menggerakan sayapnya dan terbang. Bila seekor singa menerima pesan yang sama, dia akan menggerakkan kakinya dan berlari. Keduanya menjawab sebuah pesan yang sama, namun yang sesuai dengan kemampuan hewan tersebut. Ini disebut polimorfisme karena sebuah variabel tungal dalam program dapat memegang berbagai jenis objek yang berbeda selagi program berjalan, dan teks program yang sama dapat memanggil beberapa metode yang berbeda di saat yang berbeda dalam pemanggilan yang sama. Hal ini berlawanan dengan bahasa fungsional yang mencapai polimorfisme melalui penggunaan fungsi kelas-pertama.

**Langkah Kerja :**

1. Buatlah 2 Program dengan tema buah bagi nomor urut ganjil dan jika genap maka tema hewan.

# BAB II

# PEMBAHASAN

1. Program Buah Pertama

```
#include <iostream>
using namespace std;
int b;
class buahNaga{
	public:
		buahNaga(int);
		int totBeli();
		int jumMatang();
	private:
		int beli;
		int matang;
};
buahNaga::buahNaga(int b){
	beli=b;
	matang=b;
}
buahNaga::totBeli(){
	return beli;
}
buahNaga::jumMatang(){
	return matang;
}
int main(){
	buahNaga bl(3);
	buahNaga mt(3);
	cout << "============================="<< endl;
	cout << "||Membeli Buah Naga           ||"<< endl;
	cout << "============================="<< endl;
	cout << "||Total Buah Yang Di Beli : " <<bl.totBeli()<<endl;
	cout << "||Jumlah Buah Yang Matang : " <<mt.jumMatang()<<endl;
	return 0;
}
```

Dari program di atas pada bagian di bawah ini

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

Fungsi #include<iostream> adalah agar kita dapat menggunakan file header seperti perintah ( cout, cin, endl, ). Sedangkan “using namespace std; “ digunakan untuk memanggil namespace yang memiliki nama ‘std’. Namespace ‘std’ merupakan standar namespace dari C++ yang biasa kita gunakan untuk memanggil class/object/fungsi yang terdapat di dalam namespace tersebut.

```
class buahNaga{
    public:
        buahNaga(int);
        int totBeli();
        int jumMatang();
    private:
        int beli;
        int matang;
};
```

Pada program di atas dideklarasikan “class buahNaga” yang berisi “Public” dan “Private” yang dimana isi dari mode akses “public” tersebut terdiri dari buahNaga(int); , int totBeli(); dan int jumMatang(); yang akan digunakan untuk pendeklarasian constructor dan function pada program di bawahnya. Sedangkan untuk “private” akan di akses sebagai member fuction pada bagian fungsi. Seperti pada bagian di bawah ini :

```
buahNaga::buahNaga(int b){
    beli=b;
    matang=b;
}
buahNaga::totBeli(){
    return beli;
}
buahNaga::jumMatang(){
    return matang;
```

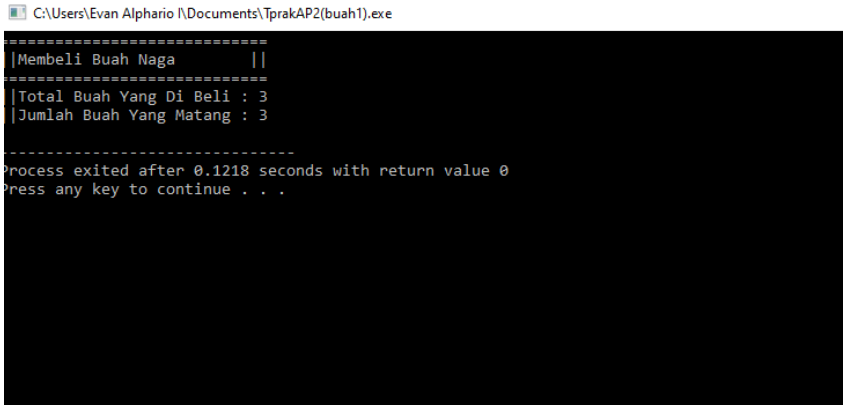
Pada bagian program di atas “buahNaga::buahNaga(int b) “ merupakan bagian constructor yang berisi private beli dan matang yang di deklarasikan sebagai variable “b”.

Pada bagian “buahNaga::totBeli()” dan “buahNaga::jumMatang()” yang dideklarasikan sebagai function dari class buahNaga yang berisi member function dari private di mana berisi tipe balikan yang akan mengembalikannya ke beli dan matang pada bagian private.

```
int main(){
	buahNaga bl(3);
	buahNaga mt(3);
	cout << "============================="<< endl;
	cout << "||Membeli Buah Naga           ||"<< endl;
	cout << "============================="<< endl;
	cout << "||Total Buah Yang Di Beli : " <<bl.totBeli()<<endl;
	cout << "||Jumlah Buah Yang Matang : " <<mt.jumMatang()<<endl;
	return 0;
}
```

Sedangkan untuk program di atas ini adalah program utama“(int main () )” Diamana pada program utama tersebut sudah di deklarasikan class dengan nama buahNaga bl dan buahNaga mt dengan nilai 3. Cout berfungsi untuk mencetak kalimat yang sudah di masukan pada program.pada cout “total buah yang di beli dan total buah yang matang” terdapat lanjutan yaitu <<bl.totBeli()<<endl; dan <<mt.jumMatang()<<endl; dimana tanda “<<” berfungsi untuk melanjutkan cout(character out) yang berisi bl.totBeli dan mt.jumBeli yang dimana mt dan bl nilai nya sudah ditentukan sebagai 3.

Dan output yang di hasilkan adalah sebagai berikut :


C:\Users\Evan Alphario I\Documents\TprakAP2(buah1).exe
||Membeli Buah Naga ||
||Total Buah Yang Di Beli : 3
||Jumlah Buah Yang Matang : 3
Process exited after 0.1218 seconds with return value 0
Press any key to continue . . .

Gambar 2.1

2. Program buah kedua.

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

int d;
class hargaBuah{
	public:
		hargaBuah(int);
		int DurianN();
		int SemangkA();
	private:
		int durian;
		int semangka;
};
hargaBuah::hargaBuah(int d){
	durian=d;
	semangka=d;
}
hargaBuah::DurianN(){
	return durian;
}
hargaBuah::SemangkA(){
	return semangka;
}
int main(){
	hargaBuah dr(15000);
	hargaBuah sm(15000);
	cout << "============================"<< endl;
	cout << "||Harga Satuan Buah-Buahan||"<< endl;
	cout << "============================"<< endl;
	cout << "||Harga Buah Durian   : " <<dr.DurianN()<<endl;
	cout << "||Harga Buah Semangka : " <<sm.SemangkA()<<endl;

	return 0;
}
```

Dari program di atas pada bagian di bawah ini

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

Fungsi #include<iostream> adalah agar kita dapat menggunakan file header seperti perintah ( cout, cin, endl, ). Sedangkan “using namespace std; “ digunakan untuk memanggil namespace yang memiliki nama ‘std’. Namespace

‘std’ merupakan standar namespace dari C++ yang biasa kita gunakan untuk memanggil class/object/fungsi yang terdapat di dalam namespace tersebut.

```
class hargaBuah{
	public:
		hargaBuah(int);
		int DurianN();
		int SemangkA();
	private:
		int durian;
		int semangka;
};
```

Pada program di atas dideklarasikan “class hargaBuah” yang berisi “Public” dan “Private” yang dimana isi dari mode akses “public” tersebut terdiri dari hargaBuah (int); , int Durian(); dan int Semangka(); yang akan digunakan untuk pendeklarasian constructor dan function pada program di bawahnya. Sedangkan untuk “private” akan di akses sebagai member fuction pada bagian fungsi. Seperti pada bagian di bawah ini :

```
hargaBuah::hargaBuah(int d){
	durian=d;
	semangka=d;
}
hargaBuah::DurianN(){
	return durian;
}
hargaBuah::SemangkA(){
	return semangka;
}
```

Pada bagian program di atas “hargaBuah::hargaBuah(int d)“ merupakan bagian constructor yang berisi private durian dan semangka yang di deklarasikan sebagai variable “d”.

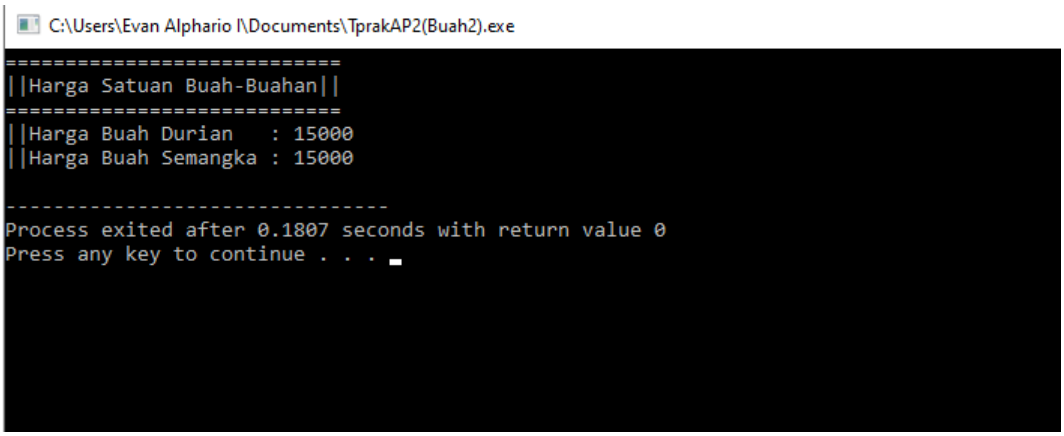
Pada bagian “hargaBuah::DurianN()” dan “hargaBuah::SemangkaA()” yang dideklarasikan sebagai function dari class hargaBuah yang berisi member function dari private di mana berisi tipe balikan yang akan mengembalikannya ke durian dan semangka pada bagian private.

```
int main(){
	hargaBuah dr(15000);
	hargaBuah sm(15000);
	cout << "============================"<< endl;
	cout << "||Harga Satuan Buah-Buahan||"<< endl;
	cout << "============================"<< endl;
	cout << "||Harga Buah Durian   : " <<dr.DurianN()<<endl;
	cout << "||Harga Buah Semangka : " <<sm.SemangkA()<<endl;

	return 0;
}
```

Sedangkan untuk program di atas ini adalah program utama“(int main () )” Diamana pada program utama tersebut sudah di deklarasikan class dengan nama hargaBuah dr dan hargaBuah sm dengan nilai 15000. Cout berfungsi untuk mencetak kalimat yang sudah di masukan pada program.pada cout “total buah yang di beli dan total buah yang matang” terdapat lanjutan yaitu <<dr.DurianN()<<endl; dan <<sm.SemangkA()<<endl; dimana tanda “<<” berfungsi untuk melanjutkan cout(character out) yang berisi dr.DurianN dan sm.SemangkA yang dimana mt dan bl nilai nya sudah ditentukan sebagai 15000.

Dan output yang di hasilkan adalah sebagai berikut :




Gambar 2.2

# BAB III
# KESIMPULAN

Dari program di atas dapat di simpulkan bahwa **#Include<iostream>** berfungsi untuk penggunaan header seperti perintah ( cout, cin, endl, ). Sedangkan **Class** berfungsi sebagai suatu unit yang terdiri atas definisi-definisi data dan fungsi-fungsi yang menunjuk pada berbagai macam isi yang bersangkutan dengan nama pada class tersebut. Dan fungsi return(balikan) adalah untuk membalikan nilai pada titik yang di tentukan misalkan pada bagian class yang dimana ia akan me return ke titik yang di tentukan pada bagian class, sama hal nya dengan return pada bagian program utama. Cout berfungsi untuk menampilkan/mencetak kalimat yang sudah dimasukan pada bagian program yang diawali dengan cout. Endl; berfungsi untuk membuat garis baru / pindah baris agar kalimat tidak menyatu dengan kalimat cout yang di cetak selanjutnya.

**DAFTAR PUSTAKA**

Tim Dosen Teknik Informatika UPR,2020.Pemrograman Berorientasi Objek

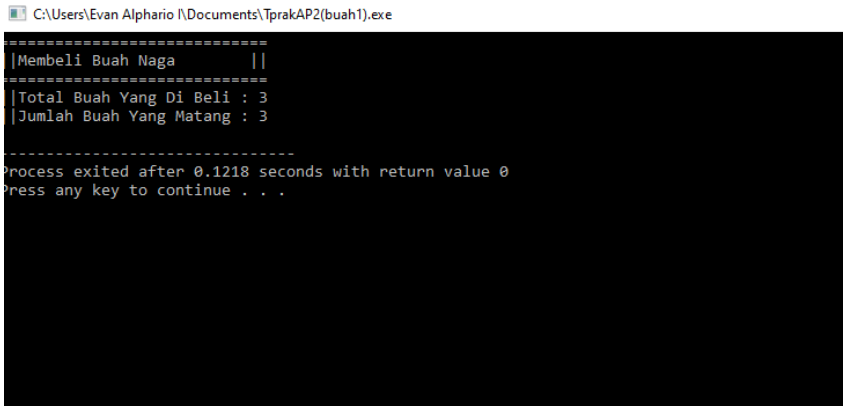
NoName,2015.Compiler File Header dan Fungsinya https://almuhtadi93.wordpress.com/2015/01/25/compiler-file-header-dan-fungsinya-using-namespace-std-cara-membuat-komentar-integer-string-dan-char/.

Pascal,2014.Macam-macam file header dan fungsinya di bahasa c++ http://pascaldhika.blogspot.com/2014/03/macam-macam-file-header-dan-fungsinya_22.html.

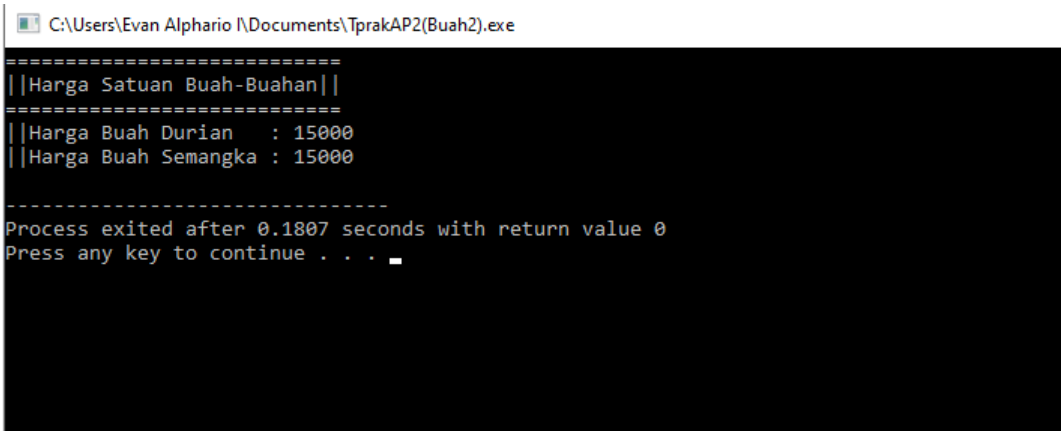
Unknown,2013.Pengertian PBO http://hadiprojek.blogspot.com/2013/03/pemrograman-berorientasi-objek.html.

diperbaikan lagi!

# LAMPIRAN




Gambar 5.1




Gambar 5.2