

6

45

14

3

:

3

---

71

**LAPORAN HASIL PRAKTIKUM**  
**ALGORITMA DAN PEMROGRAMAN II**



COVER  
tidak  
sesuai

**NAMA** : Bryant Aprilian Bahan  
**NIM** : DBC 118 100  
**KELAS** : A  
**MODUL** : I (DASAR PEMROGRAMAN  
BERORIENTASI OBJEK)

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

2020

# BAB I

## TUJUAN DAN LANDASAN TEORI

### A. Tujuan

1. Memahami dasar-dasar pemrograman berorientasi obyek
2. Memahami enkapsulasi
3. Membuat kelas dan objek

### B. Landasan Teori

Pemrograman berorientasi objek (*object-oriented programming* disingkat OOP) merupakan pemograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigma ini dibungkus dalam kelas-kelas atau objek-objek. Setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya.

#### 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.

1. **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 menggerakkan 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 tunggal 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.

Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah kita tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut (terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Sebagai contoh anggap kita memiliki sebuah departemen yang memiliki manager, sekretaris, petugas administrasi data dan lainnya. Misal manager tersebut ingin memperoleh data dari bag administrasi maka manager tersebut tidak harus mengambilnya langsung tetapi dapat menyuruh petugas bag administrasi untuk mengambilnya. Pada kasus tersebut seorang manager tidak harus mengetahui bagaimana cara mengambil data tersebut tetapi manager bisa mendapatkan data tersebut melalui objek petugas administrasi. Jadi untuk menyelesaikan suatu masalah dengan kolaborasi antar objek-objek yang ada karena setiap objek memiliki deskripsi tugasnya sendiri.

tidak menggunakan langkah kerja

## BAB II LANGKAH KERJA

### A. LANGKAH KERJA

1. Buatlah 2 program dengan tema buah bagi yang nomor ganjil dan jika nomor genap maka tema hewan dengan menggunakan prinsip dasar pemrograman berorientasi objek !

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

class hewan{
    public:
        hewan(int);
        int jlhMata();
        int jlhKaki();

    private:
        int mata;
        int kaki;
};

hewan::hewan(int i){
    mata=i;
    kaki=i;
}

int hewan::jlhMata(){
    return mata;
}

int hewan::jlhKaki(){
    return kaki;
}
```

```
int main() {  
    hewan m(2);  
    hewan h(1);  
    cout<<"===== Hewan Anjing =====<<endl;  
    cout<<"jumlah Mata anjing adalah :"<<m.jlhMata()<<endl;  
    cout<<"jumlah Kaki anjing adalah :"<<h.jlhKaki();  
  
    return 0;  
}
```

## BAB III

### PEMBAHASAN

Pembahasan langkah kerja pada Laporan Akhir Praktikum Modul 1 *Pemrograman Berorientasi Objek* ini sebagai berikut:

#### 1. Program Pertama

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

`#include` merupakan satu jenis pengarah *preprocessor* yang digunakan untuk membaca/mengakses *file* judul (*header file*). Preprocessor selalu dijalankan terlebih dahulu pada saat proses kompilasi terjadi. `<iostream.h>` dan `<conio.h>` merupakan *file header* yang merupakan *standard library* dari C++.

```
class hewan{
    public:
        hewan(int);
        int jlhMata();
        int jlhKaki();
}
```

`hewan` merupakan class yang akan digunakan pada program ini, atau bisa dikatakan sebagai main class program. nama class merupakan bentuk penyederhanaan dari suatu permasalahan yang berkaitan dengan objek. Maka dari itu kelas dapat didefinisikan sebagai sesuatu yang mempunyai data (sifat) dan fungsi (kelakuan).

Sebuah kelas menggunakan kata kunci yang merupakan pemberian hak akses terhadap data-data di dalam kelas itu sehingga disebut dengan *access specifier*.

```
public:

    hewan(int);

    int jlhMata();

    int jlhKaki();


private:

    int mata;

    int kaki;
```

Setelah memasukkan nama class, kemudian ada statement dimana perintah *public* menyatakan bahwa perintah-perintah yang ada di bawahnya dapat diakses *diluar* class.

*Private* termasuk access specifier sama seperti Public. Akan tetapi, *private* memberi pengertian bahwa perintah yang ada dibawahnya hanya dapat diakses di dalam class tersebut.

Pendeklarasian variabel dari class pada potongan program diatas digunakan untuk pemanggilan nilai yang nanti akan digunakan pada prosedur selanjutnya. Syntax *int* adalah tipe data integer atau bilangan bulat.

```
int hewan::jlhMata(){

    return mata;

}


int hewan::jlhKaki(){

    return kaki;

}
```



```

int main(){
    hewan m(2);
    hewan h(1);
    cout<<"===== Hewan Anjing ====="<<endl;
    cout<<"jumlah Mata anjing adalah :"<<m.jlhMata()<<endl;
    cout<<"jumlah Kaki anjing adalah :"<<h.jlhKaki();

    return 0;
}

```

*cout*. *cout* berfungsi untuk menampilkan keluaran atau output ke layar monitor.

Tanda << (dua buah tanda kurang dari berurutan) merupakan sebuah operator yang disebut operator “penyisipan/peletakan”. Operator ini akan mengarahkan operand (data) yang terletak di sebelah kanannya obyek yang terletak di sebelah kiri. *Cout* juga dapat melakukan output secara beruntun. Artinya, satu penulisan *cout* dapat diikuti oleh banyak operator.

*Endl* merupakan suatu fungsi manipulator yang digunakan untuk menyisipkan karakter NewLine atau mengatur pindah baris. File header yang harus disertakan adalah file header *iostream.h*.

Baris ini mengatakan kepada kompiler bahwa ada sebuah fungsi bernama *main*, yang mana fungsi itu memiliki nilai balik bertipe data integer, sehingga diberi tanda *int*. Tanda kurung { dan } menandakan awal dan akhir fungsi dalam program diatas dan menghentikan kode lainnya disebut tubuh fungsi. Lebih spesifik lagi, semua yang terletak di dalam tanda {} disebut blok. Tanda { menyatakan awal eksekusi program, dan } merupakan akhir eksekusi program.

Statemen **return 0** yang berisi nilai 0 berfungsi untuk mengembalikan nilai ke system operasi.

## 2. Program Ketiga

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

class hewan{
    public:
        hewan(int);
        int jlhTelinga();
        int jlhBuntut();

    private:
        int telinga;
        int buntut;
};

hewan::hewan(int t){
    telinga=t;
    buntut=t;
}
```

```
int hewan::jlhTelinga(){
    return telinga;
}

int hewan::jlhBuntut(){
    return buntut;
}

int main(){
    hewan m(2);
    hewan h(1);
    cout<<"===== Hewan Babi ====="<<endl;
    cout<<"jumlah telinga babi adalah :"<<m.jlhTelinga()<<endl;
    cout<<"jumlah buntut babi adalah :"<<h.jlhBuntut();

    return 0;
}
```

dijelaskan!

Paragraft

## **BAB IV**

### **KESIMPULAN**

- Pemrograman Berorientasi Objek (PBO) menggambarkan dinamika kelas dan objek dalam suatu program aplikasi membuat program semakin mendekati apa yang terjadi di dunia nyata (Simulasi).
- Header berfungsi sebagai library yang tanpanya beberapa fungsi tidak kan berjalan.
- Penulisan badan program diawali dengan “{“ dan diakhiri dengan “}”.
- Class berfungsi sebagai identitas badan utama yang memiliki karakteristik tersendiri dan memiliki hak akses berbeda-beda (public/private/protected).
- C++ memiliki banyak operator.

## **BAB V**

### **DAFTAR PUSTAKA**

Dosen TI. 2020. *Modul Praktikum Algoritma dan Pemrograman II*.  
Jurusan Teknik Informatika Fakultas Teknik Universitas Palangka Raya

Agus Seputra. 2013. Pemrograman Berorientasi Objek pada C++.  
[http://www.agusseputra.com/2013/05/pemrograman-berorientasi-objek-pada-c\\_4784.html](http://www.agusseputra.com/2013/05/pemrograman-berorientasi-objek-pada-c_4784.html) (diakses pada tanggal 06 April 20120 pukul 14:30 WIB)

Salman Fariz. 2012. Bab 2: Elemen-Elemen Di Dalam C++  
<https://salmanfz.wordpress.com/2012/02/25/element-elemen-di-dalam-c2/>  
(diakses pada tanggal 06 April 2020 pukul 15:12 WIB)

selinliz. 2008. pengertian bahasa pemrograman c++  
<https://selinliz.wordpress.com/2008/09/11/pengertian-bahasa-pemograman-c/> (diakses pada tanggal 06 April 2020 pukul 15:57 WIB)

diperhatikan lagi!!!

## BAB VI

### LAMPIRAN

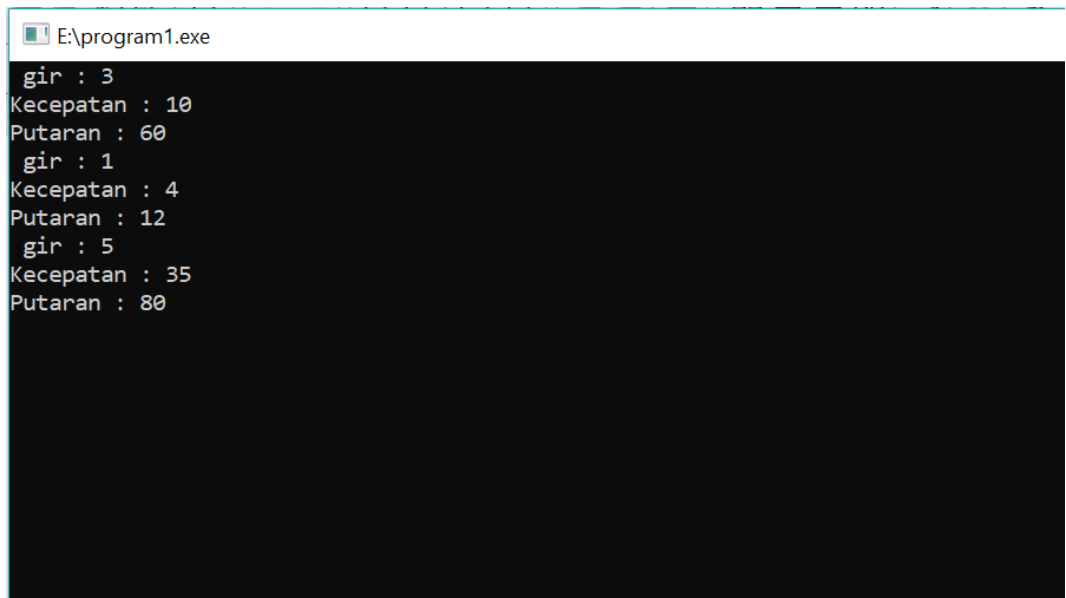
- Source Code Program 1

```
Modul1Tugas.NIM118032.cpp  program1.cpp
1  #include<iostream>
2  #include<conio.h>
3  using namespace std;
4  class Sepeda {
5  public:
6      Sepeda (int, int, int);
7      void mengubahPutaran(int);
8      void mengubahGir(int);
9      void mengorek();
10     void tampilInfo();
11 private:
12     int kecepatan;
13     int putaran;
14     int gir;
15 };
16
17 Sepeda::Sepeda(int k, int p, int g)
18 {
19     kecepatan = k;
20     putaran = p;
21     gir = g;
22 }
23
24 void Sepeda::mengubahPutaran (int p)
25 {
26     putaran = p;
27 }
28
29 void Sepeda::mengubahGir(int g)
30 {
31     gir = g;
32 }
33
34 void Sepeda::mengorek ()
35 {
36     cout<<"Kecepatan dan Putaran berkurang ....";
37 }
38
39 void Sepeda::tampilInfo()
40 {
41     cout<<" gir : "<<gir<<endl<<"Kecepatan : "<<kecepatan<<endl<<"Putaran : "<<putaran<<endl;
42 }
43
44 int main ()
45 {
46     Sepeda Sepeda1(10,60,1),
47     Sepeda2(5,12,5),
48     Sepeda3(35,80,5);
49     Sepeda1.tampilInfo();
50     Sepeda2.tampilInfo();
51     Sepeda3.tampilInfo();
52
53     getch();
54     return 0;
55 }
```

- Source Code Program 3

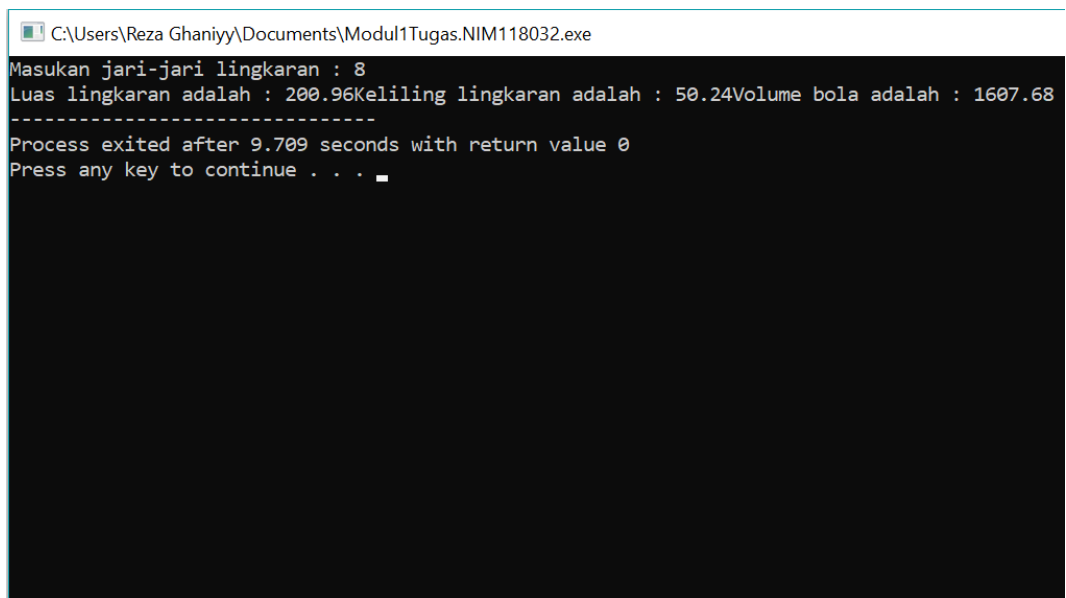
```
Modul1Tugas.NIM118032.cpp
1  #include<iostream>
2  using namespace std;
3  int main ()
4  {
5      int r,d;
6      float phi=3.14, luas, keliling, volume;
7
8      cout<<"Masukan jari-jari lingkaran : ";
9      cin>>r;
10
11      d = r+r;
12      luas = phi*r*r;
13      keliling = phi*d;
14      volume = 4/3*phi*r*r*r;
15
16      cout<<"Luas lingkaran adalah : "<<luas;
17      cout<<"Keliling lingkaran adalah : "<<keliling;
18      cout<<"Volume bola adalah : "<<volume;
19
20      return 0;
21 }
```

- **Output Program 1**



```
E:\program1.exe
gir : 3
Kecepatan : 10
Putaran : 60
gir : 1
Kecepatan : 4
Putaran : 12
gir : 5
Kecepatan : 35
Putaran : 80
```

- **Output Program 3**



```
C:\Users\Reza Ghaniyy\Documents\Modul1Tugas.NIM118032.exe
Masukan jari-jari lingkaran : 8
Luas lingkaran adalah : 200.96Keliling lingkaran adalah : 50.24Volume bola adalah : 1607.68
-----
Process exited after 9.709 seconds with return value 0
Press any key to continue . . .
```