

**LAPORAN HASIL PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN II**



**NAMA : IRWANDI**  
**NIM : 193030503054**  
**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 : Irwandi  
NIM : 193030503054  
Kelas : A  
Modul : Dasar Pemrograman Berorientasi Objek

Komposisi	MAX	Nilai
BAB I Tujuan dan Landasan Teori	10	6
BAB II Pembahasan	60	55
BAB III Kesimpulan	20	10
Daftar Pustaka	5	3
Lampiran	5	3
Jumlah	100	77

Penilai

Asisten Pratikum

Diana

# **BAB I**

## **TUJUAN DAN LANDASAN TEORI**

### **I. Tujuan**

Setelah menyelesaikan modul ini, mahasiswa diharapkan mampu

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

### **II. Landasan Teori**

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

Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar. Lebih jauh lagi, pendukung OOP mengklaim bahwa OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat.

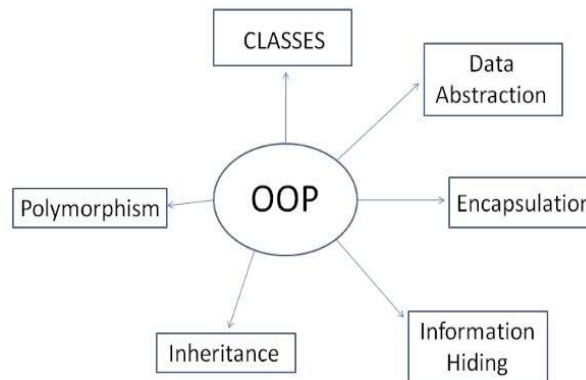
#### **Kelas**

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.

mengenal lebih dalam tentang Pemrograman Berorientasi Object atau disingkat (PBO) atau englishnya (OOP) Object Oriented Programming, alangkah lebih baiknya kita lihat penjelasan Pemrograman Berorientasi Object berikut:



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.

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 pesan tersebut dikirim.

## BAB II

### PEMBAHASAN

#### 2.1. Program Pertama

```
#include<iostream>
#include<string.h>
using namespace std;
class Hewan
{ public:
    string Says() { return "?"; }
};
class Kucing: public Hewan
{ public:
    string Says() { return "Meong..."; }
};
int main()
{ Kucing* c = new Kucing(); //Create object Kucing
  Hewan* h = c; //Pointer Hewan refer to Kucing
  cout << "Suara Kucing : " << h->Says() << endl; //call method on
  Kucing
  cout << "Suara Hewan : " << h->Says() << endl; //depends on virtual
  return 0;
}
```

Contoh program 1 class hewan

#### Pengenalan Praprosesor #Include

#include merupakan salah satu jenis pengarah praprosesor (preprocessor directive). Pengarah praprosesor ini dipakai untuk membaca file yang diantaranya

berisi deklarasi fungsi dan definisi konstanta. File-file ini mempunyai ciri yaitu

namanya diakhiri dengan ekstensi .h.

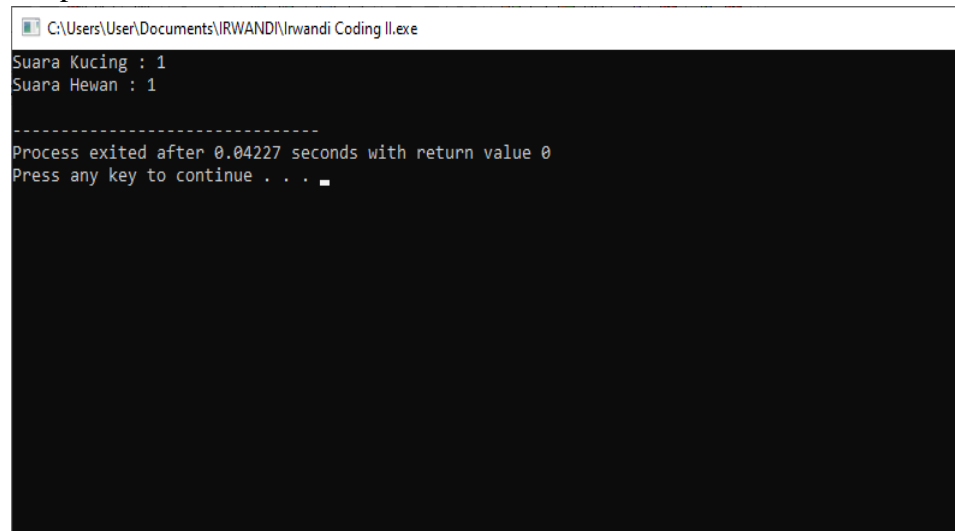
y1.5

1. 

#Include<Nama File>
  
2. 

#Include "Nama File"

### Output :



Hasil Output Program 1 class Hewan

Bentuk pertama(`#include<namafile>`)mengisyaratkan bahwa pencarian file dilakukan pada direktori khusus, yaitu direktori file include. Sedangkan bentuk kedua (`#include "namafile"`)menyatakan bahwa pencarian file dilakukan pertama kali pada direktori aktif tempat program sumber dan seandainya tidak ditemukan pencarian akan dilanjutkan pada direktori lainnya yang sesuai dengan perintah pada sistem operasi.

### Komentar Dalam Program

Untuk keperluan dokumentasi dengan maksud agar program mudah dipahami di saat lain, biasanya pada program disertakan komentar atau keterangan mengenai program. Dalam C, suatu komentar dapat ditulis dengan dua cara:

1. Ditulis dengan diawali dengan tanda `/* ....*/`. Untuk penulisan dengan tanda ini, penulisan dapat lebih dari satu baris. Awal komentar diberi tanda `/*` dan akhir komentar diberi tanda `*/`
2. Ditulis dengan tanda `//`. Penulisan dengan tanda ini hanya untuk satu baris komentar. Tanda `//` diberikan diawal komentar saja.

### Penjelasan :

Program diatas memperlihatkan yaitu string says, (, ), { return" ?" ; class, fublic, string says {return, Meong dan } int main() pointer, cout<<"suara kucing : "<<h->says()<<endl; //call method on kucing cout<<"suara hewan : "<<h-Says()<<endl; return dan endl adalah suatu keyword Token = dan << adalah operator Token(, ), {, :, dan }

```

class Kucing: public Hewan
{
public:
    string Says() { return "Meong..."; }
};

int main()
{
    Kucing* c = new Kucing(); //Create object Kucing
    Hewan* h = c; //Pointer Hewan refer to Kucing
    cout << "Suara Kucing : " << h->Says() << endl; //call method on
    Kucing
    cout << "Suara Hewan : " << h->Says() << endl; //depends on
    virtual
    return 0;
}

```

#### .Class

Class merupakan cetak biru (blue print) dari objek atau dengan kata lain sebuah Class menggambarkan ciri-ciri objek secara umum. Sebagai contoh Suzuki Smash, Yamaha VegaR, Honda SupraFit, dan Kawasaki KazeR merupakan objek dari Class sepeda motor. Suzuki Smash dan objek lainnya juga memiliki kesamaan atribut (merk, tipe, berat, kapasitas bensin, tipe mesin, warna, harga) dan method untuk mengakses data pada atributnya (misal fungsi untuk menginputkandatamerk, tipe, berat, dsbsertafungsi untuk mencetakdatamerk,tipe, berat, dsb).

```
namaClass namaObjek = new namaClass()
```

Class utama dari program :

```

int main()
{
    Kucing* c = new Kucing(); //Create object Kucing
    Hewan* h = c; //Pointer Hewan refer to Kucing
    cout << "Suara Kucing : " << h->Says() << endl; //call method on
    Kucing
    cout << "Suara Hewan : " << h->Says() << endl; //depends on
    virtual
    return 0;
}

```

Perhatikanclass Main diatas!

Namaobjekdari class hewan/kucing.

Silahkan dicobauntuk melihat hasilnya

#### Catatan:

Semua data yang diinputkan dianggap sebagai suatu nilai String meskipun data tersebut hanya terdiri atas angka saja. Untuk menampung data yang diinputkan ke dalam variabel bertipe numerik (misal :string, int main,), maka data harus terlebih dahulu diubah ke tipe data numerik.

#### 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. 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.

#### MethodOverriding

Overriding method adalah kemampuan dari subclass untuk memodifikasi method dari superclass-nya, yaitu dengan cara menumpuk (mendefinisikan kembali) method superclass-nya. Contoh overriding method dapat dilihat pada class-class turunan dari class Bentuk yang mendefinisikan kembali method gambar() dan method hapus() dari class induknya.

Method juga adalah kumpulan program yang mempunyai nama. Program harus dibungkus dalam method. Dengan method kita bisa memanggil kumpulan program hanya dengan memanggil nama methodnya, pekerjaan jadi lebih singkat dan tidak boros menuliskan program, program menjadi lebih terstruktur, praktis, dan efisien. Bentuk umum: <nama\_method>(<parameter/argument>); // menggunakan tanda kurung setelah nama method itu kuncinya // parameter/argument bersifat opsional, tergantung kebutuhan

#### Return ekspresi;

Dengan ekspresi adalah sebuah ekspresi yang nilainya dinyatakan untuk sebuah variable yang tipenya sama seperti tipe return. Terdapat juga fungsi yang tidak memberikan nilai return atau tipe returnnya void.



## 2.2. Program Kedua

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

class Hewan
{ public:
    string Says() {return "1";}
};
class Anjing: public Hewan
{ public:
    string Says() { return "menggonggong"; }
};
int main()
{   Anjing* c = new Anjing(); //Create object Anjing
    Hewan* h = c;  //Pointer Hewan refer to Anjing

    cout << "Suara Anjing : " << h->Says() << endl; //call method
on Anjing
    cout << "Suara Hewan : " << h->Says() << endl; //depends
on virtual
    return 0;
}
```

Contoh program 2 class Hewan

### Pengenalan Praprosesor #Include

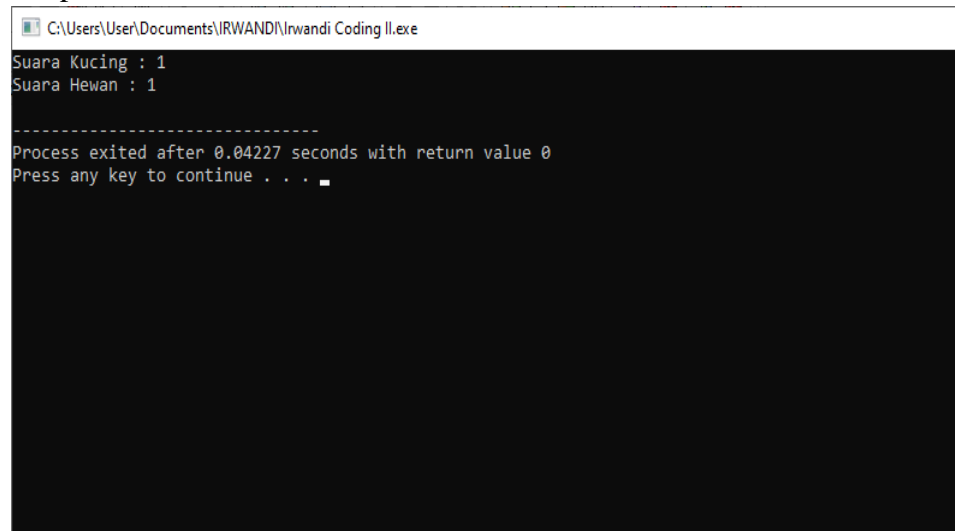
#include merupakan salah satu jenis pengarah praprosesor (preprocessor directive). Pengarah praprosesor ini dipakai untuk membaca file yang diantaranya berisi deklarasi fungsi dan definisi konstanta. File-file ini mempunyai ciri yaitu namanya diakhiri dengan ekstensi .h.

1. 

#Include<Nama File>
2. 

#Include "Nama File"

### Output :



```
C:\Users\User\Documents\IRWANDI\Irwindi Coding II.exe
Suara Kucing : 1
Suara Hewan : 1
-----
Process exited after 0.04227 seconds with return value 0
Press any key to continue . . .
```

Hasil Output Program 2 class Hewan

Bentuk pertama(`#include<namafile>`)mengisyaratkan bahwa pencarian file dilakukan pada direktori khusus, yaitu direktori file include. Sedangkan bentuk kedua (`#include "namafile"`)menyatakan bahwa pencarian file dilakukan pertama kali pada direktori aktif tempat program sumber dan seandainya tidak ditemukan pencarian akan dilanjutkan pada direktori lainnya yang sesuai dengan perintah pada sistem operasi.

### Komentar Dalam Program

Untuk keperluan dokumentasi dengan maksud agar program mudah dipahami di saat lain, biasanya pada program disertakan komentar atau keterangan mengenai program. Dalam C, suatu komentar dapat ditulis dengan dua cara:

1. Ditulis dengan diawali dengan tanda `/* ....*/`. Untuk penulisan dengan tanda ini, penulisan dapat lebih dari satu baris. Awal komentar diberi tanda `/*` dan akhir komentar diberi tanda `*/`
2. Ditulis dengan tanda `//`. Penulisan dengan tanda ini hanya untuk satu baris komentar. Tanda `//` diberikan diawal komentar saja.

### Penjelasan :

Program diatas memperlihatkan yaitu string says, (, ), { return" ?" ; class, fublic, string says {return, Menggonggong dan } int main() pointer, cout<<"suara kucing : "<<h->says()<<endl; //call method on kucing cout<<"suara hewan : "<<h-Says()<<endl; return dan endl adalah suatu keyword Token = dan << adalah operator Token(, ), {, :, dan }

```

class Hewan
{ public:
    string Says() {return "1";}
};
class Anjing: public Hewan
{ public:
    string Says() { return "menggonggong"; }
};
int main()
{   Anjing* c = new Anjing(); //Create object Anjing
    Hewan* h = c;   //Pointer Hewan refer to Anjing

    cout << "Suara Anjing : " << h->Says() << endl; //call method
on Anjing
    cout << "Suara Hewan : " << h->Says() << endl; //depends
on virtual
    return 0;
}

```

#### .Class

Class merupakan cetak biru (blue print) dari objek atau dengan kata lain sebuah Class menggambarkan ciri-ciri objek secara umum. Sebagai contoh Suzuki Smash, Yamaha VegaR, Honda SupraFit, dan Kawasaki KazeR merupakan objek dari Class sepeda motor. Suzuki Smash dan objek lainnya juga memiliki kesamaan atribut (merk, tipe, berat, kapasitas bensin, tipe mesin, warna, harga) dan method untuk mengakses data pada atributnya (misal fungsi untuk menginputkandatamerk, tipe, berat, dsbsertafungsi untuk mencetakdatamerk,tipe, berat, dsb).

```
namaClass namaObjek = new namaClass()
```

Class utama dari program :

```

int main()
{   Anjing* c = new Anjing(); //Create object Anjing
    Hewan* h = c;   //Pointer Hewan refer to Anjing

    cout << "Suara Anjing : " << h->Says() << endl; //call method
on Anjing
    cout << "Suara Hewan : " << h->Says() << endl; //depends on
virtual
    return 0;
}

```

Perhatikan class Main di atas!  
Nama objek dari class hewan/kucing.  
Silahkan dicoba untuk melihat hasilnya

#### Catatan:

Semua data yang diinputkan dianggap sebagai suatu nilai String meskipun data tersebut hanya terdiri atas angka saja. Untuk menampung data yang diinputkan ke dalam variabel bertipe numerik (misal : string, int main, ), maka data harus terlebih dahulu diubah ke tipe data numerik.

#### 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. 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.

#### Method Overriding

Overriding method adalah kemampuan dari subclass untuk memodifikasi method dari superclass-nya, yaitu dengan cara menumpuk (mendefinisikan kembali) method superclass-nya. Contoh overriding method dapat dilihat pada class-class turunan dari class Bentuk yang mendefinisikan kembali method gambar() dan method hapus() dari class induknya.

Method juga adalah kumpulan program yang mempunyai nama. Program harus dibungkus dalam method. Dengan method kita bisa memanggil kumpulan program hanya dengan memanggil nama methodnya, pekerjaan jadi lebih singkat dan tidak boros menuliskan program, program menjadi lebih terstruktur, praktis, dan efisien. Bentuk umum: <nama\_method>(<parameter/argument>); // menggunakan tanda kurung setelah nama method itu kuncinya // parameter/argument bersifat opsional, tergantung kebutuhan

#### Return ekspresi;

Dengan ekspresi adalah sebuah ekspresi yang nilainya dinyatakan untuk sebuah variable yang tipenya sama seperti tipe return. Terdapat juga fungsi yang tidak memberikan nilai return atau tipe returnnya void.

# inti sari

## BAB III KESIMPULAN

### 1.3. Kesimpulan

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. 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.

mengenal lebih dalam tentang Pemrograman Berorientasi Object atau disingkat (PBO) atau englishnya (OOP) Object Oriented Programming, alangkah lebih baiknya kita lihat penjelasan Pemrograman Berorientasi Object berikut:

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.

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 pesan tersebut dikirim.

## BAB IV

### DAFTAR FUSTAKA

#### 1.4 Daftar Fustaka

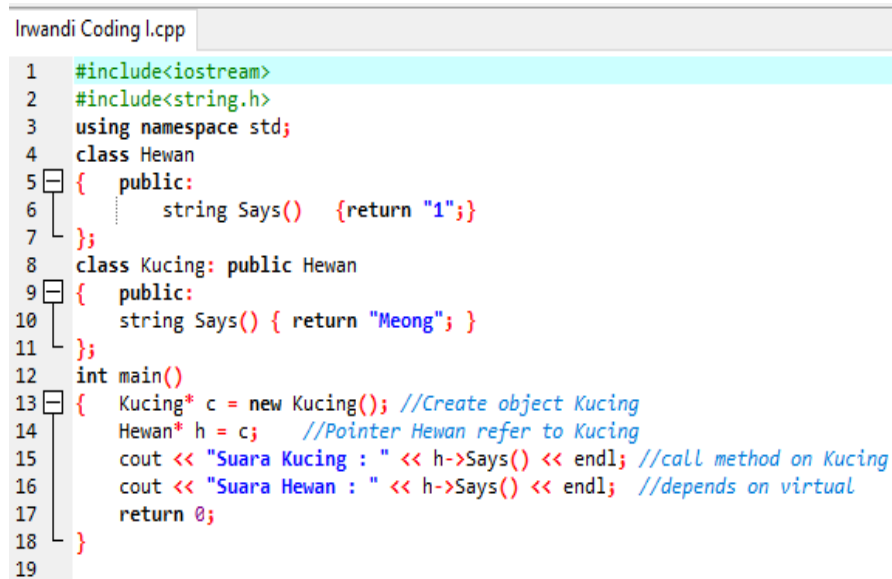
1. Tim Asisten Dosen. 2013. *Modul 3 INHERITANCES* .Malang: Universitas Negeri Malang.
2. <http://www.mediatutorial.web.id/2013/12/inheritance-antar-class-dalam-cplusplus.html>

## BAB V

### LAMPIRAN

#### 1.5 Lampiran

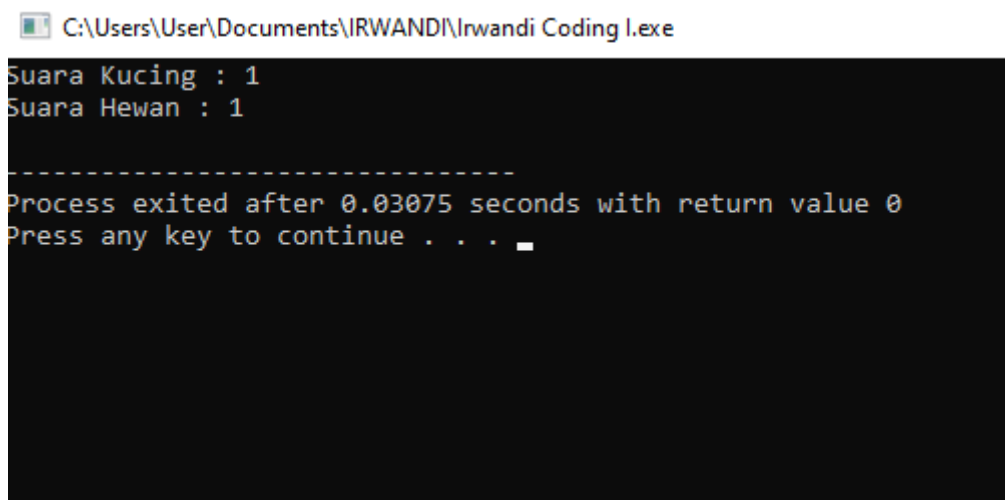
##### 1. Program 1 Class Hewan



```
Irwandi Coding I.cpp
1  #include<iostream>
2  #include<string.h>
3  using namespace std;
4  class Hewan
5  { public:
6      string Says() {return "1";}
7  };
8  class Kucing: public Hewan
9  { public:
10     string Says() { return "Meong"; }
11 };
12 int main()
13 {   Kucing* c = new Kucing(); //Create object Kucing
14     Hewan* h = c; //Pointer Hewan refer to Kucing
15     cout << "Suara Kucing : " << h->Says() << endl; //call method on Kucing
16     cout << "Suara Hewan : " << h->Says() << endl; //depends on virtual
17     return 0;
18 }
19
```

Gambar Program 1 Class Hewan

Output :



```
C:\Users\User\Documents\IRWANDI\Irwandi Coding I.exe
Suara Kucing : 1
Suara Hewan : 1

-----
Process exited after 0.03075 seconds with return value 0
Press any key to continue . . .
```

Gambar Output Program 1 Class Hewan

## 2. Program 2 Class Hewan

```
Inwandi Coding II.cpp
1  #include<iostream>
2  #include<string.h>
3  using namespace std;
4
5  class Hewan
6  { public:
7      string Says() {return "1";}
8  };
9  class Anjing: public Hewan
10 { public:
11     string Says() { return "menggonggong"; }
12 };
13 int main()
14 { Anjing* c = new Anjing(); //Create object Anjing
15   Hewan* h = c; //Pointer Hewan refer to Anjing
16
17   cout << "Suara Anjing : " << h->Says() << endl; //call method on Anjing
18   cout << "Suara Hewan : " << h->Says() << endl; //depends on virtual
19   return 0;
20 }
21
```

Gambar Program 2 Class Hewan

Output :

```
C:\Users\User\Documents\IRWANDI\sembarangan.exe
Suara Anjing : 1
Suara Hewan : 1

-----
Process exited after 0.09291 seconds with return value 0
Press any key to continue . . .
```

Output Program 2 Class Hewan



