

LAPORAN HASIL PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN II



NAMA : SUGENG WAHYU NUGROHO
NIM : 193010503005
KELAS : A
MODUL : II (PEWARISAN)

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PALANGKA RAYA
2020

LAPORAN HASIL PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN II



NAMA : SUGENG WAHYU NUGROHO

NIM : 193010503005

KELAS : A

MODUL : II (PEWARISAN)

Komposisi	MAX	Nilai
BAB I Tujuan dan Landasan Teori	10	8
BAB II Pembahasan	60	49
BAB III Kesimpulan	20	10
Daftar Pustaka	5	5
Lampiran	5	4
Jumlah	100	76

Penilai

Asisten Praktikum

DIANA

BAB I

TUJUAN DAN LANDASAN TEORI

1.1. TUJUAN

Setelah menyelesaikan modul ini, mahasiswa diharapkan mampu membuat kelas baru dari kelas yang sudah ada dengan pewarisan.

1.2. DASAR TEORI

Dalam PBO, kita mengambil realita kehidupan sehari-hari. Kita melakukan pengamatan bahwa manusia secara alami sering melakukan pengelompokan atas objek atau benda. Sejauh ini kita mengetahui cara untuk melakukan pengelompokan-pengelompokan atas objek-objek yang serupa (menjadi kelas objek).

Selain melakukan kategorisasi terhadap objek yang memiliki sekumpulan atribut dan perilaku yang sama, manusia sering melakukan pengelompokan terhadap objek yang memiliki kesamaan atas beberapa (tidak semua) atribut/perilaku. Contoh: Pengelompokan atas kendaraan bermotor, kemudian menggrupkannya berdasarkan suatu tipe atau jenis (mobil, truk, sepeda motor, dll.). Setiap subkategori ini merupakan kelas atas objek-objek yang serupa.

Ada beberapa karakteristik yang di-share oleh semua kelompok. Relasi antar kelas-kelas ini disebut dengan relasi “is-a”. Dalam setiap kasus, objek yang dikelompokkan bersama dalam satu sub-kategori merupakan anggota dari kategori yang lebih umum. Contohnya adalah seperti di bawah ini.

- a. Mobil adalah (“is-a”) kendaraan bermotor.
- b. Truk adalah (“is-a”) kendaraan bermotor
- c. Sepeda Motor adalah (“is-a”) kendaraan bermotor

Objek yang dikelompokkan dalam satu kelas men-share sekumpulan atribut dan perilaku. Jadi, seluruh objek kendaraan bermotor memiliki sekumpulan atribut dan perilaku yang juga dimiliki oleh objek dari mobil. Keterkaitan antar kelas dalam relasi “is-a” berasal dari kenyataan bahwa sub

kelas memiliki atribut dan perilaku yang dimiliki kelas induk, ditambah atribut dan perilaku yang dimiliki oleh sub kelas tersebut.

Superclass (“kelas dasar” atau “kelas induk”) merupakan kelas yang lebih general dalam relasi “is-a”. Subclass (“kelas turunan” atau “kelas anak”) merupakan kelas yang lebih spesifik dalam relasi “is-a”. Objek yang dikelompokkan dalam sub kelas memiliki atribut dan perilaku kelas induk, dan juga atribut dan perilaku tambahan. (Jadi, kumpulan atribut dan perilaku sub kelas lebih besar dari super kelas-nya). Relasi “is-a” antar superclass dan subclasses-nya disebut dengan pewarisan atau inheritance.

Subclass “mewarisi” suatu superclass (atau juga bisa dikatakan sebuah subclass “turunan dari” suatu superclass) karena reusabilitas Perangkat Lunak, membuat kelas baru (kelas turunan) dari kelas yang sudah ada (kelas dasar), kelas turunan mewarisi kelas induk yang mendapatkan data dan perilaku, merupakan bentuk spesial dari kelas induk, dan diperluas dengan perilaku tambahan,

Pewarisan ada dua jenis yaitu pewarisan tunggal dan pewarisan jamak. Pada protected access, protected members dapat diakses oleh member kelas dasar, friend kelas dasar, member kelas turunan, dan friend kelas turunan. Kelas turunan dapat merujuk/mengakses langsung public dan protected data member kelas induk dengan menggunakan nama atribut yang yang diakses.

```
#include<iostream.h>
#include<conio.h>
class Point {
public:
    Point(float = 0.0, float = 0.0);
    void cetakPoint();
protected: float x,y;
};
```

```

Point::Point(float a, float b)
{
    cout<< "Konstruktor Point dijalankan "<<endl;
    x = a;
    y = b;
} void Point::cetakPoint() {
    cout<< "Point : "<< '['<<x<< ", "<<y<< ']'<<endl;
}
class Circle : public Point {
public:
    Circle(float r = 0.0, float a = 0.0, float b =
0.0); //konstruktor
    float area();
    void cetakPoint();
protected:
    float radius,l;
};
Circle::Circle(float r, float a, float b): Point(a, b){
    cout<< "Konstruktor Circle dijalankan"<<endl; radius = r;
} float Circle::area()
{l=3.14*radius*radius; return l;
}
void Circle::cetakPoint()
{ Point::cetakPoint(); cout<< "Circle dgn r : "<<radius<< "
Center : ["<<x<< ", "<<y<< "]"<<endl;
}

```

```

        cout<< "Luas circle : "<<l<<endl;
    }
    class Cylinder:public Circle {
    public:
        Cylinder(float h = 0.0, float r = 0.0, float a = 0.0, float b
= 0.0); //konstrukto
        float area();
        float vol();
        void cetakPoint();
    protected:
        float height,l,v; };
    Cylinder::Cylinder(float h, float r, float a, float b) : Circle(r, a,
b) {
        cout<< "Konstruktor Cylinder dijalankan"<<endl;
        height = h; }
    float Cylinder::area(){
        l= (2*Circle::area() + 2*3.14*radius*height); return l; }
    float Cylinder::vol() {
        v= (Circle::area()*height); return v; }
    void Cylinder::cetakPoint()
    {
        Circle::cetakPoint();
        cout<< "Tinggi tabung (h) : "<<height<<endl;
        cout<< "Luas tabung : "<<l<<endl;
        cout<< "Volume tabung : "<<v<<endl;
    }
    int main() {

```

```

{
    Point p(1.1, 2.2);

    Cout<<endl;

    Circle lingkaran (10, 5, 5);

    Lingkaran.area();

    Lingkaran.cetakPoint();

    Cout<<endl;

    Cylinder tabung(20, 30, 3, 4);

    Tabung.area();

    Tabung.vol();

    Tabung.cetakPoint();

    Getch();

    Return 0;
}

```

Dalam kelas pada pemrograman C++, terdapat mekanisme keturunan (*inheritance*), dimana dalam mekanisme ini terdapat istilah kelas induk dan kelas anak. Kelas anak akan menginduk pada kelas induk. Keistimewaan dari metode *inheritance* adalah, kelas anak memiliki sifat-sifat atau fungsi dari kelas induk, namun dengan “batasan” tertentu. Dengan mekanisme keturunan ini, member kelas anak akan bertambah yaitu member kelas anak sendiri ditambah dengan member kelas induk.

Sebagai contoh, kelas anak memiliki member a dan b, sedangkan kelas induk memiliki member c. Apabila kelas anak adalah keturunan dari kelas

induk, maka member kelas anak adalah a, b, dan c. Namun di sini kelas induk tidak berhak akan member anak yaitu a dan b.

Kamu dapat menyatakan bahwa suatu kelas adalah keturunan dari kelas lainnya menggunakan sintaks berikut ini:

```
class nama_kelas_anak:penentu_akses nama_kelas_induk
{
    //kode
}
```

Pada contoh sintaks di atas, nama_kelas_anak adalah kelas turunan dari kelas induk. Kemudian nama_kelas_induk adalah nama kelas induk dari si-anak. Nah, disini yang paling penting yaitu penentu_akses, dimana penentu akses adalah hak akses maksimal dari member kelas induk yang diwariskan kepada kelas anak. Berikut adalah macam-macam penentu_akses yang mempengaruhi pewarisan/ *inheritance*:

- a. *Public* : maka member kelas induk yang berupa *public* akan tetap *public* dan yang *protected* tetap *protected* pada kelas anak.
- b. *Protected* : maka member kelas induk yang berupa *public* akan berubah menjadi *protected*, dan *protected* tetap sebagai *protected*.
- c. *Private* : maka member yang diturunkan dari kelas induk pada kelas anak semua berubah menjadi *private*, entah itu *public* atau *protected*.

Agar kamu memiliki gambaran tentang penentu akses, simak contoh-contoh berikut. Jika kamu memiliki kelas induk sebagai berikut:

```
class induk
{
    private:
        int member_Private1, member_Private2;
```



```
protected:

    int member_Protected1, member_Protected2;

public:

    int member_Public1, member_Public2;

};
```

Dan kelas anak dideklarasikan sebagai berikut:

```
Class anak : public induk

{

Public:

    Int member Public_Anak;

};
```

Maka, dalam kelas anak, `member_Protected1`, dan `member_Protected2`, akan tetap *protected* dan `member_Public1` dan `member_Public2` akan tetap *public*. Sementara kelas anak tetap tidak bisa mendapatkan `member_Private1` dan `member_Private2`.

Jika kamu mendeklarasikan kelas anak seperti berikut:

```
class anak: protected induk

{

    public:

        int member_Public_Anak;

};
```

Maka dalam kelas anak, `member_Protected1` dan `member_protected2`, akan tetap *protected* sedangkan `member_Public1` dan `member_Public2` akan berubah menjadi *protected*.

Jika kamu mendeklarasikan kelas anak seperti berikut:

```
class anak: private induk
{
    public:
        int member_Public_Anak;
};
```

Maka dalam kelas anak, `member_Protected1`, `member_Protected2`, `member_Public1`, dan `member_Public2` akan berubah semuanya menjadi *private* jika kamu mengakses melalui kelas anak.

Implementasi *inheritance* pada classes, Mimin contohkan sebagai berikut:

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

class induk{
protected:
    int sisi_a,sisi_b;
public:
    void input(float panjang, float lebar);
};

void induk::input(float panjang, float lebar){
    sisi_a = panjang;
    sisi_b = lebar;
}
```

```

class anak:public induk{
public:
float luas(){return sisi_a*sisi_b;};
};

int main(){
anak o;
o.input(7,8);
cout<<"Nilai dari o.luas() adalah = "<<o.luas()<<endl;
return 0;
}

```

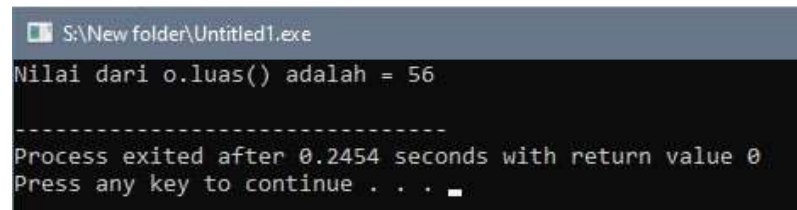
Pada program di atas kita membuat dua buah kelas yaitu kelas induk dan kelas anak. Kelas induk bertugas dalam menerima input nilai dari user, sedangkan kelas anak berfungsi dalam proses kalkulasi nilai yang didapatkan dari input user tadi.

Kelas induk memiliki member *protected* berupa sisi_a dan sisi_b serta member *public* berupa fungsi input. Fungsi input bertugas menerima input nilai melalui parameter panjang serta lebar yang dimilikinya dan selanjutnya diumpkan pada member *protected* sisi_a serta sisi_b.

Kelas anak dideklarasikan terhadap kelas induk dengan penentu akses berupa *public*. Kelas anak ini memiliki member *public* berupa fungsi luas. Seperti penjelasan di atas, ketika kelas anak dideklarasikan dengan penentu akses *public*, maka member induk *protected* akan tetap *protected* dan *public* akan tetap *public*. Pada baris 18, bisa kamu lihat bahwa, kelas anak mengakses member induk yaitu sisi_a dan sisi_b, dimana member ini bersifat *protected*.

Pada operasinya di fungsi utama/ main(), kelas anak membuat objek 'o'. Kemudian objek ini digunakan dalam mengakses member fungsi input milik kelas induk serta diberikan masukkan nilai 7 dan 8. Sebagai hasil operasi, objek 'o' kemudian mengakses member fungsi luas miliknya sendiri dan mencetak hasil kalkulasi pada baris 24.

Berikut output program yang kita dapatkan:

A screenshot of a Windows command prompt window. The title bar at the top reads "S:\New folder\Untitled1.exe". The command prompt shows the output of a program: "Nilai dari o.luas() adalah = 56". This is followed by a line of dashes "-----". Below the dashes, it says "Process exited after 0.2454 seconds with return value 0". The final line is "Press any key to continue . . .", with a small black square cursor at the end of the line.

```
S:\New folder\Untitled1.exe
Nilai dari o.luas() adalah = 56
-----
Process exited after 0.2454 seconds with return value 0
Press any key to continue . . .
```

Gambar 2.1 tampilan output program

BAB II

PEMBAHASAN



~~3.1.~~ Pembahasan Tugas Modul II Nomor 1

Program yang dibuat adalah program untuk memberikan informasi yang berkaitan dengan buah.

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

Pada 2 baris pertama, akan mendeklarasikan *library* agar program yang akan dibuat dapat dijalankan sebagaimana mestinya. Library *iostream* berfungsi untuk menampilkan sintaks input dan output program seperti *cin* dan *cout*. Library *String.h* adalah library yang berisi fungsi-fungsi untuk membantu pengolahan string ataupun substring. Fungsi-fungsi tersebut antara lain : *strcpy*, *strncpy*, *strcat*, *strncat*, *strcmp*, *strncmp*, *strlen*. Didalam program ini saya akan menambahkan sintaks *strcpy* yang berfungsi untuk menyalin nilai dari sebuah variable string (string asal) ke variable string tujuan.

Bentuk umum:

```
Strcpy (var_tujuan, string_asal);
```

Lalu selanjutnya..

```
Using namespace std;

Classs Buah
{
    Protected:
    Int jlhdaun;

    Char nama[100];Char warna[100];
```

Sintaks *using namespace std;* berfungsi untuk mempersingkat penulisan kode yang semula “*std::cin*” , “*std::cout*” atau “*std::endl*” menjadi hanya “*cin*”, “*cout*”, dan “*endl*”. Fungsi utama *using namespace std* pada pendeklarasian tersebut adalah untuk memberitahukan kepada kompiler bahwa kita akan menggunakan semua fungsi , class atau file yang terdapat pada memori namespace std. Jadi jika anda telah menggunakan pernyataan “*using namespace std*” kita tidak perlu repot-repot menambahkan *std::* di depan fungsi-fungsi untuk memanggil fungsi seperti fungsi *cout*, *cin* dan sebagainya yang terdapat dalam namespace std.

Setelah itu saya membuat class dengan nama Buah. Lalu penggunaan sintaks *protected:* berfungsi agar semua anggota yang berada dalam lingkungan *protected* hanya dapat diakses oleh kelas yang mendefinisikannya dan kelas turunannya (sub-class).

Int jlhdau;, penggunaan tipe data *int* dimaksudkan karena jumlah daun nanti pastinya bernilai bilangan bulat. *Char nama[100]* dan *char warna[100]* tipe data *char* digunakan karena nama dari data nanti pasti berupa huruf dan dalam kutung 100 mendeklarasikan jumlah karakter yang dapat diinput pada program.

```
public:

    //konstruktor

    Buah()

    {

        cout<<" Konstruktor Buah dijalankan "<<endl;

        cout<<" -----\n";

    }
```

Public, public yang artinya badan program ini yang akan diakses oleh program secara umum. Dalam badan program public disinilah badan private akan diakses, dimana dalam *// Pembentuk atau outputan yang diberikan* akan mengisi data yang dimasukan dalam program ke tempatnya.

Pembentukan konstruktor setelah public, program pada bagian ini adalah yang pertama dibaca oleh sistem, diawali dengan mendeklarasikan dahulu nama dari Class induknya yaitu *Buah()*. Sengaja mencantumkan “*konstruktor Buah dijalankan*” agar dapat dibuktikan bahwa program ini yang pertama dieksekusi. *Cout<<”-----”*, agar dapat memudahkan dalam membaca program nanti.

```
//setter

Void setJlhdaun(int jlhdaun)
{
    This -> jlhdaun = jlhdaun;
}

Void setNama(char n[20])
{
    Strcpy(nama,n);
}

voidsetWarna(char w[20])
{
    Strcpy(warna,w);
}
```

Setter, digunakan untuk memberikan set nilai untuk suatu properti. function ini sering digunakan untuk memvalidasi data yang masuk sebelum data tersebut diisikan pada suatu properti. Sedangkan getter digunakan untuk menghasilkan suatu nilai dari hasil perhitungan.

Jika kita ingin mendeklarasikan variabel, maka harus diawali dengan “set” didepannya karena mendeklarasikan di setter. Contoh pada program di atas adalah `setjlhdaun(int jlhdaun)`, ini mendeklarasikan pada variabel yang sudah dituliskan sebelumnya. Selanjutnya untuk getter..

```
//getter  
  
Int getjlhdaun()  
{  
  
    return jlhdaun = jlhdaun;  
  
}  
  
Char *getnama()  
{  
  
    Return nama;  
  
}  
  
Char *getwarna()  
{  
  
    Return warna;  
  
}  
};
```


Setelah setter dideklarasikan selanjutnya kita memasukkan nilai yang terkandung pada setter tadi dengan diawali “*get”.

```
//class turunan
Class apel: public Buah
{
Private:
    Int daun;
Public:
    //konstruktor
    apel()
    {
        Cout<<"|////////////////////////////////////////|\n";
        Cout<<"| -:: CIRI BUAH APEL ::-|\n";
        Cout<<"|////////////////////////////////////////|\n";<<endl;
    }
    Void setdaun(int d)
    {
        Daun = d;
    }
    Int getdaun()
    {
        return daun;
    }
};
```

Class apel:public Buah, adalah class turunan yang disini saya beri nama apel, titid dua berarti menandakan bahwa class ini berasal dari class public induknya. Untuk *private* nya adalah hanya int daun saja. Daun dideklarasikan sebagai d, dan int get dimasukan. Return daun agar sistem membaca ini berasal dari variabel daun.

```

Int main()

{

    apel apelku;

    apelku.setNama("Apel");

    cout<<"|Nama    =    "<<apelku.getnama()<<"\t\t
|"<<endl;

    apelku.setJlhdaun(2);

    cour<<"|Daun = "<<apelku.getjlhdaun()<<"\t\t |"<<endl;

    apelku.setWarna("Merah");

    cout<<"|Warna = "<<apelku.getwarna()<<"\t\t |"<<endl;

    cout<<" _____ "

    cin.get();

    system("Pause");

}

```

int main(), pada sintaks ini lah program akan di proses, program yang akan disekeksi harus dibuat pemanggilnya dulu. Disini saya benama *apelku.setnama* yang nantinya output yang akan keluar adalah "*Apel*". *Apelku.setJlhdaun(2)* yang nantinya input yang ditampilkan adlaah dari vaeiabel jumlah daun yang didalam kurung adalah jumlah daunnya yaitu 2. *Apelku.setwarna("Merah")* yang nantinya akan ditampilkan dari variabel warna.disini saya beri keternagan warna "*Merah*".

System("Pause"), ditambahkan agar saat di akhir program atau saat ingin keluar dari program, akan ditampilkan info konfirmasi "*Press any key to continue...*", yang menyuruh kita menekan tombol lain untuk keluar.

Maka tampilan kedua program adalah..

```
1  #include <iostream>
2  #include <string.h>
3
4  using namespace std;
5
6  class Buah
7  {
8  protected:
9
10     int jlhdaun;
11     char nama[100];
12     char warna[100];
13
14 public:
15     //konstruktor
16     Buah(){
17         cout<<" Konstruktor Buah dijalankan "<<endl;
18         cout<<" -----\\n";
19     }
20
21     //setter
22     void setJlhdaun(int jlhdaun)
23     {
24         this -> jlhdaun = jlhdaun;
25     }
26     void setName(char n[20])
27     {
28         strcpy(nama,n);
29     }
30     void setWarna(char w[20])
31     {
32         strcpy(warna,w);
33     }
34     //getter
35     int getJlhdaun()
36
37     {
38         return jlhdaun;
39     }
40     char *getnama()
41     {
42         return nama;
43     }
44     char *getwarna()
45     {
46         return warna;
47     }
48 };
49
50 //class turunan
51 class apel:public Buah
52 {
53 private:
54     int daun;
55 public:
56     //konstruktor
57     apel()
58     {
59         cout<<"|//////////\\n";
60         cout<<"|  -::CIRI BUAH APEL:-  |\\n";
61         cout<<"|//////////\\n"<<endl;
62     }
63     void setdaun(int d)
64     {
65         daun = d;
66     }
67     int getdaun()
68     {
```

```

int getdaun()
{
    return daun;
}
};
int main()
{
    apel apelku;
    apelku.setNama("Apel");
    cout<< "Nama    = "<<apelku.getnama()<<"\t\t    |"<<endl;
    apelku.setJlhdaun(2);
    cout<< "Daun    = "<<apelku.getjlhdaun()<<"\t\t    |"<<endl;
    apelku.setWarna("Merah");
    cout<< "Warna    = "<<apelku.getwarna()<<"\t\t    |"<<endl;
    cout<< "_____";
    cin.get();
    system("Pause");
}

```

Gambar 3.1 program pertama

```

S:\pakaibaru\apel.exe
Konstruktor Buah dijalankan
-----
|////////////////////////|
|  - :: CIRI BUAH APEL :: -  |
|////////////////////////|
|
|Nama    = Apel          |
|Daun    = 2             |
|Warna    = Merah        |
|
|_____

```

Gambar 3.2 output program pertama

3.2. Pembahasan program kedua

Untuk program kedua ini hampir sama dengan program pertama tadi, yang membedakan hanyalah variabel-variabel nya saja.

```

S:\pakaibaru\mangga.exe
Konstruktor Buah dijalankan
-----
|////////////////////////|
|      Ciri Mangga Buah      |
|////////////////////////|
|
|Nama    = Mangga         |
|Daun    = 3              |
|Warna    = Hijau         |
|
|_____

```

Gambar 3.3 output program pertama

BAB III

KESIMPULAN

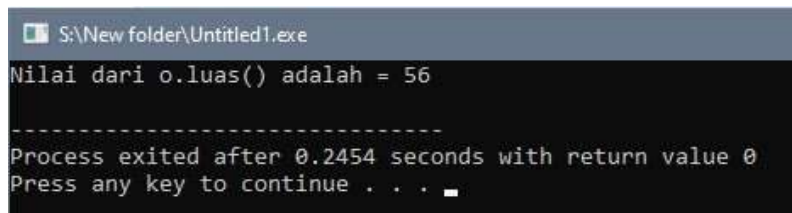
Inheritance atau pewarisan memungkinkan variabel atau fungsi anggota dari suatu kelas diwariskan kelas lain. Kelas yang mewariskan disebut *kelas dasar*, sedangkan kelas yang diwarisi disebut *kelas turunan*.

Pewarisan ada dua jenis yaitu pewarisan tunggal dan pewarisan jamak

DAFTAR PUSTAKA


- Febrian, R. (2014). *Memahami Fungsi Setter Getter Pada Penerapan Encapsulation*. Indonesia:
<https://www.dumetschool.com/blog/Memahami-Fungsi-Setter-Getter-Pada-Penerapan-Encapsulation>.
- Koding, A. (2018). *PENGERTIAN DAN FUNGSI STRING*. Indonesia:
<http://abangkoding.blogspot.com/2018/08/string-bahasa-c.html>.
- Mahasiswa. (2016). *Rizka Reza Pahlevi*. Indonesia:
http://jagocoding.com/tutorial/920/Penggunaan_Private_dan_Public_dalam_C.
- Santoso, R. (2017). *C++ - Inheritance (Dasar Pewarisan)*. Indonesia:
<http://www.nblognlife.com/2017/04/c-inheritance-dasar-pewarisan.html>.
- Unknow. (2017). *Kegunaan Using namespace std;*. Indonesia:
<http://rzkyiff.blogspot.com/2017/03/kegunaan-using-namespace-std.html>.

LAMPIRAN



```
S:\New folder\Untitled1.exe
Nilai dari o.luas() adalah = 56
-----
Process exited after 0.2454 seconds with return value 0
Press any key to continue . . .
```

Gambar 2.1 tampilan output program



```
1  #include <iostream>
2  #include <string.h>
3
4  using namespace std;
5
6  class Buah
7  {
8  protected:
9
10     int jlhdaun;
11     char nama[100];
12     char warna[100];
13
14 public:
15     //konstruktor
16     Buah(){
17         cout<<" Konstruktor Buah dijalankan "<<endl;
18         cout<<" -----\\n";
19     }
20
21     //setter
22     void setJlhdaun(int jlhdaun)
23     {
24         this -> jlhdaun = jlhdaun;
25     }
26     void setName(char n[20])
27     {
28         strcpy(nama,n);
29     }
30     void setWarna(char w[20])
31     {
32         strcpy(warna,w);
33     }
34     //getter
35     int getJlhdaun()
```



```
S:\pakaibaru\apel.exe
Konstruktor Buah dijalankan
-----
|////////////////////////|
|  -:CIRI BUAH APEL:-  |
|////////////////////////|
|
|Nama   = Apel          |
|Daun   = 2             |
|Warna  = Merah         |
|_____
```

Gambar 3.2 output program pertama

```

34 //getter
35 int getjlhdaun()
36 {
37     return jlhdaun;
38 }
39 char *getnama()
40 {
41     return nama;
42 }
43 char *getwarna()
44 {
45     return warna;
46 }
47 };
48
49 //class turunan
50 class apel:public Buah
51 {
52 private:
53     int daun;
54 public:
55     //konstruktor
56     apel()
57     {
58         cout<<"|////////////////////////|\n";
59         cout<<"|  -:CIRI BUAH APEL:-  |\n";
60         cout<<"|////////////////////////|\n"<<endl;
61     }
62     void setdaun(int d)
63     {
64         daun = d;
65     }
66     int getdaun()
67     {
68         return daun;
69     }
70 };
71
72 int main()
73 {
74     apel apelku;
75     apelku.setNama("Apel");
76     cout<<"|Nama    = "<<apelku.getnama()<<"\t\t    |"<<endl;
77     apelku.setJlhdaun(2);
78     cout<<"|Daun    = "<<apelku.getjlhdaun()<<"\t\t    |"<<endl;
79     apelku.setWarna("Merah");
80     cout<<"|Warna    = "<<apelku.getwarna()<<"\t\t    |"<<endl;
81     cout<<"|_____|";
82     cin.get();
83     system("Pause");
84 }

```

Gambar 3.1 program pertama

```

Construktör Buah dijalankan
|////////////////////////|
|      Ciri Mangga Buah      |
|////////////////////////|
|Nama    = Mangga          |
|Daun    = 3                |
|Warna    = Hijau          |
|_____|

```

Gambar 3.3 output program pertama