

LAPORAN HASIL PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN



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

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PALANGKA RAYA
2020

LAPORAN HASIL PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN



NAMA : SUGENG WAHYU NUGROHO

NIM : 193010503005

KELAS : A

MODUL : II (POLIMORFISME)

| Komposisi | MAX | Nilai |
|---------------------------------|-----|-------|
| BAB I Tujuan dan Landasan Teori | 10 | 8 |
| BAB II Pembahasan | 60 | 48 |
| BAB III Kesimpulan | 20 | 10 |
| Daftar Pustaka | 5 | 5 |
| Lampiran | 5 | 5 |
| Jumlah | 100 | |

Penilai

Asisten Praktikum

DIANA

BAB I

TUJUAN DAN LANDASAN TEORI

1.1 TUJUAN

Setelah menyelesaikan modul ini, mahasiswa diharapkan mampu membuat polimorfisme.

1.2 DASAR TEORI

Polimorfisme merupakan fitur pemrograman berorientasi obyek yang penting setelah pengkapsulan (encapsulation) dan pewarisan (inheritance). Polimorfisme berasal dari bahasa Yunani, *poly*(banyak) dan *morphos* (bentuk). Polimorfisme menggambarkan kemampuan kode-kode bahasa C++ berperilaku berbeda tergantung situasi pada waktu run (program berjalan).

Contoh polimorfisme yang konkrit dalam dunia nyata yaitu mobil. Mobil yang ada di pasaran terdiri atas berbagai tipe dan berbagai merk, tetapi semuanya memiliki interface kemudi yang sama, seperti: stir, tongkat transmisi, pedal gas dan rem. Jika seseorang dapat mengemudikan satu jenis mobil saja dari satu merk tertentu, maka orang itu akan dapat mengemudikan hampir semua jenis mobil yang ada, karena semua mobil tersebut menggunakan interface yang sama. Harus diperhatikan disini bahwa interface yang sama tidak berarti cara kerjanya juga sama. Misal pedal gas, jika ditekan maka kecepatan mobil akan meningkat, tetapi bagaimana proses peningkatan kecepatan ini dapat berbeda-beda untuk setiap jenis mobil.

Contoh lain:



Gambar 1.2

Konstruksi polimorfisme dalam pemrograman berorientasi obyek memungkinkan untuk mengadakan ikatan dinamis (juga disebut ikatan tunda, atau ikatan akhir). Apabila fungsi-fungsi dari suatu kelas dasar didefinisikan ulang atau ditindih pada kelas turunan, maka obyek-obyek yang dihasilkan hirarki kelas berupa obyek polimorfik. Polimorfik artinya mempunyai banyak bentuk atau punya kemampuan untuk mendefinisi banyak bentuk.

Polimorfisme merupakan suatu konsep yang menyatakan bahwa sesuatu yang sama dapat memiliki berbagai bentuk dan perilaku yang berbeda. Dalam hal ini polimorfisme merupakan suatu sifat menyandarkan pada kesamaan nama dalam program. Pengenal data, instans, dan metode, bahkan nama fungsi dapat dibuat dengan nama yang sama untuk kegunaan yang berbeda.

Salah satu bentuk polimorfisme pada C++ dapat digunakan pada fungsi atau operator dan dikenal sebagai istilah overloading. Overloading terhadap fungsi akan memungkinkan sebuah fungsi dapat menerima bermacam-macam tipe dan memberikan nilai balik yang bervariasi pula.

Polimorfisme memiliki syarat-syarat sebagai berikut:

- a. Ada hirarki pewarisan.
- b. Kelas dalam hirarki pewarisan harus memiliki fungsi virtual (virtual method) dengan signature yang sama.
- c. Menggunakan pointer atau rujukan ke kelas induk. Pointer digunakan untuk memanggil fungsi virtual.

Selain overloading, Polimorfisme dapat diimplementasikan dengan menggunakan dasar function overriding (melakukan redefinisi suatu fungsi di kelas anak, fungsi yang di-override memiliki signature sama, signature sama : tipe balik, nama fungsi, parameter sama) dan pewarisan. Suatu kelas disebut abstrak apabila memiliki minimal satu fungsi abstrak. Fungsi abstrak merupakan fungsi yang tidak memiliki definisi (hanya deklarasi

fungsi)/menggunakan fungsi virtual (pure virtual). virtual balikan namaFungsi
(parameter) = 0

Contoh program:

```
#include <stdio.h>

#include <iostream>

// Deklarasi kelas-kelas

class Kotak
{
    protected :

        int panjang, lebar;

        long luas, keliling;

    public :

        Kotak (int Panjang, int Lebar);

        int ambil_panjang();

        int ambil_lebar();

        virtual void hitung(); // metode virtual

        void ambil (long& Luas, long& Keliling);

};

class Balok : public Kotak
{private :

        int tinggi;

        int volume;

    public :
```

```

Balok (int Panjang, int Lebar, int Tinggi);

        int ambil_tinggi();

        virtual void hitung(); // metode virtual

        void ambil (long& Luas, long& Keliling,
long& Volume);

};

// Definisi metode-metode kelas Kotak

Kotak::Kotak (int Panjang, int Lebar)

{

        panjang = Panjang;

        lebar = Lebar;

}

int Kotak::ambil_panjang()

{

        return panjang;

}

int Kotak::ambil_lebar()

{

return lebar;}

void Kotak::hitung(){luas = panjang * lebar;

        keliling = 2 * (panjang + lebar);

}


```

```

void Kotak::ambil (long& Luas, long& Keliling)

{

    Luas = luas;

    Keliling = keliling;}

// Definisi metode-metode kelas Balok

Balok::Balok (int Panjang, int Lebar, int Tinggi):Kotak (Panjang,
Lebar)

{

    tinggi = Tinggi;

}

int Balok::ambil_tinggi()

{

    return tinggi;

}

void Balok::hitung()

{

    Kotak::hitung();

    volume = panjang*lebar*tinggi;

}

void Balok::ambil(long& Luas, long& Keliling, long& Volume)

{

    Kotak::ambil(Luas, Keliling);

```

```

// Program Utama

int main ()
{
    Kotak A (12, 3);

    Balok B (12, 3, 8);

    std::cout << std::endl;

    std::cout << "Menghitung luas, keliling, dan volume bangun" <<
    std::endl;

    std::cout <<
    "=====
    =====" << std::endl;

    A.hitung();

    long l, k;

    A.ambil(l, k);

    std::cout << "Kotak sisi-sisinya : " << std::endl;

    std::cout << "Panjang = " << A.ambil_panjang()
    << std::endl;

    std::cout << "Lebar = " << A.ambil_lebar() <<
    std::endl;

    std::cout << "Memiliki luas " << l << " dan keliling " << k <<
    std::endl << std::endl;

```



```

long v;

        B.hitung();

        B.ambil(l, k, v);

        std::cout << "Balok dengan rusuk-rusuknya : " <<
std::endl;

        std::cout << "Panjang = " << B.ambil_panjang()
<< std::endl;

        std::cout << "Lebar = " << B.ambil_lebar() <<
std::endl;

        std::cout << "Tinggi = " << B.ambil_tinggi() <<
std::endl;

std::cout << "Memiliki luas " << l << ", keliling " << k << ", dan
volume " << v << std::endl;


        std::cout << std::endl;

```

Program di atas merupakan program untuk melakukan perhitungan luas kotak dan volume balok. Kelas-kelas yang terdapat dalam program di atas yaitu kelas kotak dan kelas balok. Pada kelas balok ini mengandung beberapa metode yang sama dengan metode yang terdapat dalam kelas balok. Dengan

menggunakan konsep polimorfisme, deklarasi metode yang sama tersebut dapat dituliskan secara lebih singkat tanpa menuliskan bagian hal-hal yang sama tersebut. Dengan kata lain, bagian metode yang dibutuhkan tersebut hanya perlu dipanggil nama metodenya saja. Perbedaan antara konsep pewarisan dan polimorfisme yaitu jika dalam pewarisan, sifat yang dimiliki oleh kelas induk diturunkan kepada kelas turunannya sehingga apabila salah satu nilai variabel yang dimiliki oleh kelas induk maupun turunan diubah maka output yang dihasilkan akan mempengaruhi semua kelas. Sedangkan dalam polimorfisme, apabila salah satu nilai variabel pada salah satu kelas, maka output yang dihasilkan tidak mempengaruhi output semua kelas, tetapi hanya pada kelas yang memuat nilai variabel tersebut.

Untuk lebih jelasnya berikut merupakan uraian serta pembahasan tentang masing-masing baris program di atas yaitu sebagai berikut :

1. Semua baris yang diawali dengan dua buah tanda slash (/), akan dianggap sebagai baris komentar dan tidak akan berpengaruh pada hasil eksekusi program. Baris komentar dipakai oleh Programmer untuk memberikan penjelasan atau keterangan tentang maksud program tersebut. Bagian dari program di atas yang menggunakan dua buah tanda slash (/) yaitu:

```
//-----  
  
//      Program contoh penerapan polimorfisme  
  
//      menggunakan Turbo C++ veri 3.0  
  
//-----  
  
//      Filename : BANGUN.CPP  
  
//-----  
  
// Deklarasi kelas-kelas  
  
...  
  
// Definisi metode-metode kelas Kotak  
  
...  
  
// Definisi metode-metode kelas Kotak  
  
...
```

2. Pernyataan yang diawali dengan tanda (#) merupakan pernyataan untuk menyertakan preprocessor. Pernyataan ini bukan untuk dieksekusi. Pernyataan seperti `#include <stdio.h>` berarti memerintahkan kompiler untuk menyertakan file header `stdio.h`. File header `stdio.h` merupakan library file input output standar. Kemudian juga terdapat fungsi `#include <iostream>`. Dalam file header ini, terdapat beberapa fungsi standar yang dipakai dalam proses input dan output. Seperti misalnya perintah `cout` yang dipakai dalam program di atas merupakan bagian dari file header `iostream`. Header yang dipanggil `<iostream>`, tanpa menggunakan “.h”. Apabila tetap menggunakan “.h” maka akan muncul peringatan (warning) ketika dieksekusi, walaupun demikian program masih tetap dapat dieksekusi dengan benar. Bagian dari program di atas yang menggunakan tanda (#) yaitu :

```
#include <stdio.h>

#include <iostream>

class Kotak
{
    protected :

        int panjang, lebar;

        long luas, keliling;

    public :

        Kotak (int Panjang, int Lebar);

        int ambil_panjang();

        int ambil_lebar();

        virtual void hitung(); // metode virtual void ambil (long &
```

3. Pada kelas Kotak, nama kelas ditulis dengan sintaks class Kotak, yang menunjukkan bahwa Kotak adalah nama suatu kelas. Tampak bahwa terdapat dua buah perubah akses (anggota kelas) yang digunakan pada kelas ini yaitu protected dan public. Yang termasuk perubah akses protected yaitu variabel panjang (bertipe data integer “int”), lebar (bertipe data integer “int”), luas (bertipe data long atau bilangan real dengan range yang lebih besar), dan keliling (bertipe data long atau bilangan real dengan range yang lebih besar). Yang termasuk perubah akses public yaitu metode Kotak, ambil_panjang() (bertipe data integer), ambil_lebar() (bertipe data integer), hitung () (bertipe virtual void), dan ambil (bertipe void). Prototype metode Kotak ini berfungsi sebagai konstruktor dan memiliki argumen yang terdiri dari variabel Panjang dan Lebar yang masing-masing bertipe integer. Metode ambil terdiri dari argumen yang terdiri dari variabel Luas dan Keliling. Prototype yang menggunakan tipe data void, berarti bahwa hasil eksekusinya tidak bertipe (tanpa parameter) dan tidak memberikan nilai balik. Prototype virtual digunakan untuk memanggil fungsi kelas induk, apabila tidak menggunakan prototype virtual ini fungsi kelas induk tidak dapat dipanggil. Definisi atas metode-metode tersebut akan dipaparkan di bagian bawah.

```
class Balok : public Kotak{  
  
private :  
  
int tinggi;  
  
int volume;  
  
public :Balok (int Panjang, int Lebar, int Tinggi);  
  
int ambil tinggi();virtual void hitung(); // metode virtual
```

4. Pada kelas Balok, nama kelas ditulis dengan sintaks `class Balok`, yang menunjukkan bahwa Balok adalah nama suatu kelas. Tampak bahwa terdapat dua buah perubah akses (anggota kelas) yang digunakan pada kelas ini yaitu `private` dan `public`. Yang termasuk perubah akses `private` yaitu variabel `tinggi` (bertipe data integer “int”) dan `volume` (bertipe data integer “int”). Yang termasuk perubah akses `public` yaitu metode `Balok`, `ambil_tinggi()`, `hitung ()` (bertipe `virtual void`), dan `ambil` (bertipe `void`). Prototype metode Balok ini berfungsi sebagai konstruktor dan memiliki argumen yang terdiri dari variabel Panjang, Lebar, dan Tinggi yang masing-masing bertipe integer. Metode `ambil` terdiri dari argumen yang terdiri dari variabel Luas, Keliling, dan Volume. Prototype yang menggunakan tipe data `void`, berarti bahwa hasil eksekusinya tidak bertipe (tanpa parameter) dan tidak memberikan nilai balik. Prototype `virtual` digunakan untuk memanggil fungsi kelas induk, apabila tidak menggunakan prototype `virtual` ini fungsi kelas induk tidak dapat dipanggil. Sebuah metode yang dideklarasikan menggunakan kata kunci `virtual` ini biasanya merupakan sebuah metode yang akan di `overloading` di kelas turunannya. Meskipun untuk pada kasus tertentu penambahan kata kunci ini tidak dibutuhkan, seperti dalam program kita ini apabila kata kunci `virtual` ini dihilangkan maka tidak terjadi masalah.

Tanda ‘&’ menunjuk kepada alamat suatu variabel. Pemberian ‘&’ dalam definisi parameter menyebabkan fungsi tidak menggandakan argumennya dalam sebuah parameter, tetapi hanya mengambil alamat variabel yang menjadi argumennya. Definisi atas metode-metode tersebut akan dipaparkan di bagian bawah.

5. Berikut ini merupakan definisi metode-metode pada kelas Kotak :

```
Kotak::Kotak (int Panjang, int Lebar)
{
    panjang = Panjang;
    lebar = Lebar;
}
int Kotak::ambil_panjang()
{
    return panjang;
}
int Kotak::ambil_lebar()
{
    return lebar;
}
void Kotak::hitung()
{
    luas = panjang * lebar;
    keliling = 2 * (panjang + lebar);
}
void Kotak::ambil (long& Luas, long& Keliling)
```

Pada metode Kotak::Kotak di atas pernyataan panjang = Panjang; berfungsi untuk memberikan inisialisasi terhadap variabel input panjang ke variabel Panjang. Demikian pula untuk pernyataan lebar = Lebar; berarti memberikan inisialisasi terhadap variabel input lebar ke variabel Lebar. Inisialisasi ini dilakukan untuk mempermudah dalam hal eksekusi baris program di bawahnya yang membutuhkan manipulasi (perhitungan) terhadap variabel-variabel tersebut.

Pada metode Kotak::ambil_panjang()terdapat ungkapan return panjang; yang berfungsi untuk mengakhiri eksekusi dan mengembalikan ke fungsi pemanggil. Ketika pernyataan return dieksekusi, eksekusi fungsi yang memuat variabel panjang segera diakhiri pada keadaan tersebut.

Pada metode Kotak::ambil_lebar()terdapat ungkapan return lebar; yang berfungsi untuk mengakhiri eksekusi dan mengembalikan ke fungsi pemanggil. Ketika pernyataan return dieksekusi, eksekusi fungsi yang memuat variabel lebar segera diakhiri pada keadaan tersebut.

Metode `Kotak::hitung()` digunakan untuk melakukan perhitungan terhadap variabel panjang dan lebar sehingga dihasilkan output berupa luas dan keliling bangun kotak. Pernyataan `luas = panjang * lebar;` berarti bahwa nilai variabel luas merupakan hasil perkalian antara variabel panjang dan variabel lebar, sehingga akan diperoleh luas bangun kotak. Sedangkan pernyataan `keliling = 2 * (panjang + lebar);` berarti bahwa nilai dari variabel keliling merupakan hasil penjumlahan variabel panjang dan variabel lebar kemudian hasilnya dikalikan dengan 2, sehingga akan diperoleh keliling bangun kotak.

Metode `Kotak::ambil (long& Luas, long& Keliling)` digunakan untuk melakukan inisialisasi terhadap isi variabel luas dan keliling. Pernyataan `Luas = luas;` digunakan untuk mengisi variabel Luas dengan nilai yang dihasilkan oleh variabel luas yang dihasilkan pada metode `Kotak::hitung()`. Sedangkan pernyataan `Keliling = keliling;` digunakan untuk mengisi variabel Keliling dengan nilai yang dihasilkan oleh variabel keliling yang dihasilkan pada metode `Kotak::hitung()`.

6. Berikut ini merupakan definisi metode-metode pada kelas Balok :

```
Balok::Balok (int Panjang, int Lebar, int Tinggi):Kotak (Panjang,
Lebar)
{
    tinggi = Tinggi;
}
int Balok::ambil_tinggi()
{
    return tinggi;
}
void Balok::hitung()
{
    Kotak::hitung();
    volume = panjang*lebar*tinggi;
}
```


Pada metode `Balok::Balok` di atas pernyataan `tinggi = Tinggi;` berfungsi untuk memberikan inisialisasi terhadap variabel input tinggi ke variabel `Tinggi`. Pada metode ini juga dibutuhkan variabel panjang dan lebar yang sebelumnya diinisialisasi dalam metode kelas `Kotak`. Hal ini menunjukkan bahwa dalam metode ini diperlukan tiga buah variabel perhitungan yaitu Panjang, Lebar, dan Tinggi.

Pada metode `int Balok::ambil_tinggi()` terdapat ungkapan `return tinggi;` yang berfungsi untuk mengakhiri eksekusi dan mengembalikan ke fungsi pemanggil. Ketika pernyataan `return` dieksekusi, eksekusi fungsi yang memuat variabel tinggi segera diakhiri pada keadaan tersebut.

Metode `Balok::hitung()` digunakan untuk melakukan perhitungan terhadap variabel panjang, lebar, dan tinggi sehingga dihasilkan output berupa luas alas, keliling alas, dan volume sebuah balok. Pernyataan `Kotak::hitung();` berarti bahwa metode ini memanggil kembali metode yang berada di dalam kelas `Kotak`, sehingga nantinya akan dihasilkan output sesuai dengan metode tersebut (luas alas dan keliling alas balok)

dalam kelas Balok. Sedangkan pernyataan `volume = panjang*lebar*tinggi;` berarti bahwa nilai dari variabel `volume` merupakan hasil perkalian antara isi variabel `panjang`, variabel `lebar` dan variabel `tinggi`, sehingga akan diperoleh volume balok.

Metode `Balok::ambil(long& Luas, long& Keliling, long& Volume)` digunakan untuk melakukan inisialisasi terhadap isi variabel `luas` alas, `keliling` alas, dan `volume` balok. Pernyataan `Kotak::ambil(Luas, Keliling);` digunakan untuk memanggil metode tersebut yang berada dalam kelas `Kotak`. Melalui metode yang dipanggil ini, maka inisialisasi terhadap variabel `luas` dan `keliling` akan sama dengan inisialisasi yang dilakukan pada kelas `Kotak`. Pernyataan `Volume = volume;` digunakan untuk mengisi variabel `Volume` dengan nilai yang dihasilkan oleh variabel `volume` yang dihasilkan pada metode `Balok::hitung()`.

7. Bagian program utama dalam keseluruhan program di atas yaitu sebagai berikut :

```
int main ()
{
    Kotak A (12, 3);

    Balok B (12, 3, 8);

    std::cout << std::endl;

    std::cout << "Menghitung luas, keliling, dan volume bangun" <<
    std::endl;

    std::cout <<
    "=====
    =====" << std::endl;

    A.hitung();

    long l, k;

    A.ambil(l, k);

    std::cout << "Kotak sisi-sisinya : " << std::endl;

    std::cout << "Panjang = " << A.ambil_panjang() << std::endl;
```

Penulisan listing program untuk bagian program utama di atas selalu diawali dengan fungsi `int main ()`. Fungsi dari program utama di atas yaitu untuk menentukan prosedur dan langkah eksekusi terhadap masing-masing kelas yang telah dibuat. Pada program utama di atas ternyata prosedur yang digunakan untuk menampilkan output ke layar yaitu tanpa meminta input dari user, artinya obyek yang dimasukkan sudah ditentukan terlebih dahulu dalam program utama.

Pernyataan Kotak A (12, 3); digunakan untuk menyatakan bahwa objek yang akan dieksekusi adalah sebuah objek A berbentuk Kotak yang mempunyai panjang = 12 dan lebar = 3. Pernyataan Balok B (12, 3, 8); digunakan untuk menyatakan bahwa objek yang akan dieksekusi adalah sebuah objek B berbentuk Balok yang mempunyai panjang = 12, lebar = 3, dan tinggi = 8.

Pernyataan seperti `std::cout` berfungsi untuk menampilkan suatu output ke layar, sedangkan pernyataan `std::endl;` berfungsi untuk menandai akhir suatu baris output (ganti baris).

Pernyataan `long l, k;` berarti mendefinisikan bahwa hasil eksekusi variabel `l` (luas) dan variabel `k` (keliling) bertipe data `long` (bilangan dengan range data yang lebih panjang). Kemudian pernyataan seperti `B.hitung();` berarti objek `B` akan dieksekusi menurut metode `hitung()` yang ada di dalam kelas `Balok`. Pernyataan seperti `B.ambil(l, k, v);` berarti objek `B` akan dieksekusi menurut metode `ambil(l, k, v)` yang ada di dalam kelas `Balok` sehingga akan dihasilkan hasil eksekusi berupa Luas, Keliling, dan Volume.

Tanda “<<” berfungsi sebagai penghubung yang diartikan untuk memasukkan data ke dalam suatu perintah eksekusi. Misalnya pernyataan `std::cout << "Lebar = " << A.ambil_lebar() << std::endl;` berarti hasil eksekusi objek `A` terhadap metode `ambil_lebar()` yang membutuhkan isi dari variabel `Lebar` akan diperintahkan untuk tampil ke layar dengan format ganti baris dari hasil eksekusi di atasnya. Demikian juga untuk baris program lainnya yang mempunyai format yang sama.

8. Tanda kurung kurawal (`{...}`) digunakan untuk menandai bagian suatu isi kelas, bagian isi metode-metode di dalam kelas (behaviour), serta bagian isi program utama.
9. Tanda titik koma (`;`) digunakan untuk menandai akhir penulisan data dan fungsi dalam program. Tanda titik koma tidak digunakan untuk mengakhiri nama kelas, nama metode, nama perubah akses, serta inisialisasi program utama. Program di atas selanjutnya disimpan dengan nama `BANGUN.CPP` pada folder yang sudah ditentukan. Untuk meng-compile sebuah file yang berisikan source code C++ menggunakan `g++`, digunakan perintah `g++` di dalam shell Unix. Untuk meng-compile source code yang disimpan dengan nama `BANGUN.CPP` digunakan perintah `~$ g++ BANGUN.CPP`. Perintah tersebut akan meng-compile `BANGUN.CPP` menggunakan compiler

g++ menjadi sebuah file executable (file yang dapat dieksekusi/dijalankan) bernama a.out. Dengan demikian untuk menampilkan hasil eksekusi program yang telah dibuat pada notepad digunakan perintah ~\$./a.out. Hasil pemanggilan dan eksekusi program di atas yang tampak pada layar yaitu sebagai berikut :

```
Menghitung luas, keliling, dan volume bangun
=====
=====

Kotak sisi-sisinya :
Panjang = 12
Lebar = 3
Memiliki luas 36 dan keliling 30
Balok dengan rusuk-rusuknya :
Panjang = 12
```

BAB II

PEMBAHASAN

2.1 Pembahasan program modul III

Tema program yang akan dibuat adalah sama dengan program yang sebelumnya, tetapi yang membedakan disini adalah program yang dibuat menggunakan prinsip polimorfisme.

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

Pada empat 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 conio.h* digunakan untuk memunculkan sintaks *getch()* yang nantinya juga akan digunakan pada program ini. *Library string* digunakan untuk memunculkan sintaks *strcpy()* pada program. Dan sintaks *windows.h* digunakan untuk memunculkan sintaks sistem(“Pause”) agar ada konfirmasi saat keluar program.

```
using namespace std;

//Kelas Dasar (Kelas Abstrak)
class Buah
{
protected:
    char nama_buah[20];
    char nama_latin[20];
    char warna_b[20];
```

Sintaks *using namespace std;* berfungsi untuk mempersingkat penulisan kode yang semula “*std::cin*”, “*std::cout*”, atau “*std::endl*” menjadi hanya “*cin*”, “*cout*”, dan “*endl*”.

Selanjutnya disini dibuat kelas abstraknya terlebih dahulu, dengan nama kelas “Buah”. Lalu protected dengan anggota char nama_buah[20], nama_latin[20], dan warna_b. Char nama digunakan untuk menampilkan karakter dan karakter yang dapat diinput maksimal hanya 20 karakter.

```
public:
    void informasi()
    {
        system("color 06");
        cout<<"\tDATA BUAH"<<endl;
        cout<<"\t-----"<<endl;
    }
    //fungsi virtual murni
    virtual void namaBuah() = 0;
    virtual void namaLatin() = 0;
    virtual void warna() = 0;
};
```

Selanjutnya pendeklaraasian public, dengan data didalamnya void dengan nama informasi. *System* (“*color 06*”) adalah sintaks yang bertujuan untuk mengganti warna baground dan dan teks. Untuk 0 adalah digunakan untuk warna hitam pada baground dan nomor 6 digunakan untuk warna teks

dengan warnaa orange. Struk yang ditampilkan pertama adalah “DATA BUAH”.

Penggunaan virtual murni ada 3 buah yang dibuat yaitu void namaBuah,void namaLatin, dan void warna.

```
//kelas Apel turunan dari kelas Buah
class Apel : public Buah
{
public:
    Apel(char* nm, char *nl, char *wb )
    {
        strcpy(nama_buah, nm);
        strcpy(nama_latin, nl);
        strcpy(warna_b, wb);
    }
    void informasi()
    {
        cout<<"\tInformasi Apel"<<endl;
    }
    void namaBuah()
    {
        cout<<"\tNama : "<<nama_buah<<endl;
    }
    void warna()
    {
        cout<<"\tWarna : \'"<<warna_b<<"\'"<<endl;
    }
    void namaLatin()
    {
        cout<<"\tLatin : "<<nama_latin<<endl;}};
```


Disini barulah membuat kelas turunanya, disini diberi nama kelas Apel turunan dari kelas buah. Untuk publicnya menggunakan pointer char nm untuk nama, pointer nl untuk nama latin dan pointer wb untuk warna buah. Setelah itu pendeklarasian masing-masing tampilan.

```
//kelas Mangga turunan dari kelas Buah
class Mangga : public Buah
{
public:
    Mangga(char* nm, char *nl, char *wb)
    {
        strcpy(nama_buah, nm);
        strcpy(warna_b, wb);
        strcpy(nama_latin, nl);
    }

    void informasi()
    {
        cout<<"\tInformasi Mangga"<<endl;
    }

    void namaBuah()
    {
        cout<<"\tNama : "<<nama_buah<<endl;
    }
}
```

Untuk pendeklarasian buah mangga sama saja dengan turunan apel tadi hanya yang diganti adalah namanya saja.

```
int main()
{
    //deklarasi objek
    Buah *obj_bnt;
    Apel apl("Apel", "Mangifera indica", "Merah");
    Mangga mng("Mangga", "Malus domestica", "Hijau");

    cout<<"\n\n"<<endl;
    cout<<"\t|-----|"<<endl;
    cout<<"\t|-.:POLIMORFISME:-|"<<endl;
    cout<<"\t|-----|\n"<<endl;

    //menunjuk ke objek dari kelas Apel
    obj_bnt = &apl;
    obj_bnt->informasi();
    obj_bnt->namaBuah();
    obj_bnt->namaLatin();
    obj_bnt->warna();

    cout<<endl;
```

Sintaks *int main()*, sintaks ini digunakan sebagai pembuka bagian utama sebuah program. Didalam sinilah nantinya program-program yang

akan dijalankan. Program diatas akan menunjuk ke masing-masing pointer yang dituju sesuai nama variabel ayng telah diinput sebelumnya. Agar meyainkan maka diberi title POLIMORFISME untuk mempercantik penampilan saat progam dijalankan.

```
//menunjuk ke objek dari kelas Mangga

obj_bnt = &mng;
obj_bnt->informasi();
obj_bnt->namaBuah();
obj_bnt->namaLatin();
obj_bnt->warna();

/*

cout<<"\tby"<<endl;
cout<<"\t :"<<endl;
cout<<"\t  sugeng wahyu nugroho"<<endl;

*/

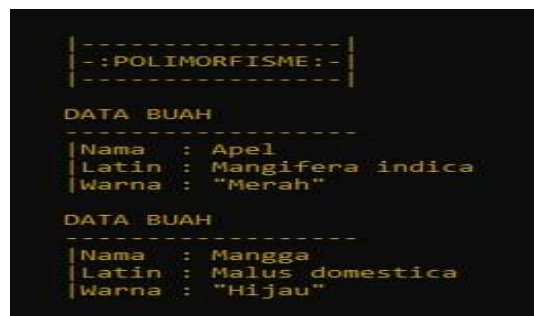
_getche();

system("Pause");

return 0;

}
```

Untuk pointer mangga sama saja penjelasannya dengan pointer apel
tampilan output program:



```
-----
-:POLIMORFISME:-
-----

DATA BUAH
-----
[Nama   : Apel
[Latin  : Mangifera indica
[Warna  : "Merah"

DATA BUAH
-----
[Nama   : Mangga
[Latin  : Malus domestica
[Warna  : "Hijau"
```

Gambar 2.1 output program

```

1  #include <iostream>
2  #include <conio.h>
3  #include <string.h>
4  #include <windows.h>
5
6  using namespace std;
7
8  //Kelas Dasar (Kelas Abstrak)
9  class Buah
10 {
11 protected:
12     char nama_buah[20];
13     char nama_latin[20];
14     char warna_b[20];
15
16 public:
17     void informasi()
18     {
19         system("color 06");
20         cout<<"\tDATA BUAH"<<endl;
21         cout<<"\t-----"<<endl;
22     }
23
24     //fungsi virtual murni
25     virtual void namaBuah() = 0;
26     virtual void namaLatin() = 0;
27     virtual void warna() = 0;
28 };
29
30 //kelas Apel turunan dari kelas Buah
31 class Apel : public Buah
32 {
33 public:
34     Apel(char* nm, char *nl, char *wb )
35     {
36         strcpy(nama_buah, nm);
37         strcpy(nama_latin, nl);
38         strcpy(warna_b, wb);
39     }
40

```

Gambar 2.2 tampilan program

```

40
41     void informasi()
42     {
43         cout<<"\tInformasi Apel"<<endl;
44     }
45
46     void namaBuah()
47     {
48         cout<<"\t>Nama : "<<nama_buah<<endl;
49     }
50
51     void warna()
52     {
53         cout<<"\tWarna : \""<<warna_b<<"\"<<endl;
54     }
55
56     void namaLatin()
57     {
58         cout<<"\t|Latin : "<<nama_latin<<endl;
59     }
60 };
61
62 //kelas Mangga turunan dari kelas Buah
63 class Mangga : public Buah
64 {
65 public:
66     Mangga(char* nm, char *nl, char *wb)
67     {
68         strcpy(nama_buah, nm);
69         strcpy(warna_b, wb);
70         strcpy(nama_latin, nl);
71     }
72
73     void informasi()
74     {
75         cout<<"\tInformasi Mangga"<<endl;
76     }
77
78     void namaBuah()
79     {

```

Gambar 2.3 tampilan program

```

79 {
80     cout<<"\t>Nama : "<<nama_buah<<endl;
81 }
82
83 void namaLatin()
84 {
85     cout<<"\t|Latin : "<<nama_latin<<endl;
86 }
87
88 void warna()
89 {
90     cout<<"\t|Warna : \"<<warna_b<<\"<<endl;
91 }
92
93
94 };
95
96 int main()
97 {
98     //deklarasi objek
99     Buah *obj_bnt;
100     Apel apl("Apel", "Mangifera indica", "Merah");
101     Mangga mng("Mangga", "Malus domestica", "Hijau");
102
103     cout<<"\n\n"<<endl;
104     cout<<"\t|-----|"<<endl;
105     cout<<"\t|:POLIMORFISME:-|"<<endl;
106     cout<<"\t|-----|\n"<<endl;
107
108     //menunjuk ke objek dari kelas Apel
109     obj_bnt = &apl;
110     obj_bnt->informasi();
111     obj_bnt->namaBuah();
112     obj_bnt->namaLatin();
113     obj_bnt->warna();
114
115
116     cout<<endl;
117
118     //menunjuk ke objek dari kelas Mangga

```

Gambar 2.4 tampilan program

```

119     cout<<endl;
120
121     //menunjuk ke objek dari kelas Mangga
122     obj_bnt = &mng;
123     obj_bnt->informasi();
124     obj_bnt->namaBuah();
125     obj_bnt->namaLatin();
126     obj_bnt->warna();
127
128     /*
129     cout<<"\tby"<<endl;
130     cout<<"\t : "<<endl;
131     cout<<"\t    sugeng wahyu nugroho"<<endl;
132     */
133     _getche();
134
135     system("Pause");
136     return 0;
137 }

```

Gambar 2.5 tampilan program

BAB III

KESIMPULAN

Polimorfisme artinya “memiliki banyak bentuk”, maksudnya satu hal yang sama dapat memiliki beberapa bentuk yang berbeda.

Analoginya, semua binatang memiliki suara maka pada kelas dasar diciptakan variabel dan fungsi untuk menampilkan suara. Namun seperti yang kita ketahui setiap jenis binatang memiliki suara yang berbeda, sehingga pada kelas turunan ditulis kembali suara yang khas dari jenis binatang tersebut.

DAFTAR PUSTAKA

- rizkyNet, A. (2011). *Tutorial Pemrograman Berorientasi Obyek dengan C++ : Polimorfisme (Studi Kasus)*. Bandung: <https://adityarizki.net/tutorial-pemrograman-berorientasi-obyek-dengan-c-polimorfisme-studi-kasus/>.
- Santoso, R. (2017). *Polimorfisme pada C++*. Malang: <http://www.nblognlife.com/2017/07/polimorfisme-pada-c.html>.
- TeknikInformatika, D. (2019). *Modul Alpro*. Palangkaraya: <http://www.nblognlife.com/2017/07/polimorfisme-pada-c.html>.

LAMPIRAN



Gambar 1.2

```

-----
-:POLIMORFISME:-
-----

DATA BUAH
-----
|Nama   : Apel
|Latin  : Mangifera indica
|Warna  : "Merah"

DATA BUAH
-----
|Nama   : Mangga
|Latin  : Malus domestica
|Warna  : "Hijau"
  
```

Gambar 2.1 output program

```

1  #include <iostream>
2  #include <conio.h>
3  #include <string.h>
4  #include <windows.h>
5
6  using namespace std;
7
8  //Kelas Dasar (Kelas Abstrak)
9  class Buah
10 {
11 protected:
12     char nama_buah[20];
13     char nama_latin[20];
14     char warna_b[20];
15
16 public:
17     void informasi()
18     {
19         system("color 06");
20         cout<<"\tDATA BUAH"<<endl;
21         cout<<"\t-----"<<endl;
22     }
23
24     //fungsi virtual murni
25     virtual void namaBuah() = 0;
26     virtual void namaLatin() = 0;
27     virtual void warna() = 0;
28 };
29
30 //kelas Apel turunan dari kelas Buah
31 class Apel : public Buah
32 {
33 public:
34     Apel(char* nm, char *nl, char *wb )
35     {
36         strcpy(nama_buah, nm);
37         strcpy(nama_latin, nl);
38         strcpy(warna_b, wb);
39     }
40
  
```

Gambar 2.2 tampilan program


```

40 |
41 | void informasi()
42 | {
43 |     cout<<"\tInformasi Apel"<<endl;
44 | }
45 |
46 | void namaBuah()
47 | {
48 |     cout<<"\tNama : "<<nama_buah<<endl;
49 | }
50 |
51 | void warna()
52 | {
53 |     cout<<"\tWarna : \""<<warna_b<<"\"<<endl;
54 | }
55 |
56 | void namaLatin()
57 | {
58 |     cout<<"\tLatin : "<<nama_latin<<endl;
59 | }
60 | };
61 |
62 | //kelas Mangga turunan dari kelas Buah
63 | class Mangga : public Buah
64 | {
65 | public:
66 |     Mangga(char* nm, char *nl, char *wb)
67 |     {
68 |         strcpy(nama_buah, nm);
69 |         strcpy(warna_b, wb);
70 |         strcpy(nama_latin, nl);
71 |     }
72 |
73 | void informasi()
74 | {
75 |     cout<<"\tInformasi Mangga"<<endl;
76 | }
77 |
78 | void namaBuah()
79 | {

```

Gambar 2.3 tampilan program

```

79 | {
80 |     cout<<"\tNama : "<<nama_buah<<endl;
81 | }
82 |
83 | void namaLatin()
84 | {
85 |     cout<<"\tLatin : "<<nama_latin<<endl;
86 | }
87 |
88 | void warna()
89 | {
90 |     cout<<"\tWarna : \""<<warna_b<<"\"<<endl;
91 | }
92 |
93 | };
94 |
95 |
96 | int main()
97 | {
98 |     //deklarasi objek
99 |     Buah *obj_bnt;
100 |     Apel apl("Apel", "Mangifera indica", "Merah");
101 |     Mangga mng("Mangga", "Malus domestica", "Hijau");
102 |
103 |     cout<<"\n\n"<<endl;
104 |     cout<<"\t|-----|"<<endl;
105 |     cout<<"\t|:-:POLIMORFISME:-|"<<endl;
106 |     cout<<"\t|-----|\n"<<endl;
107 |
108 |     //menunjuk ke objek dari kelas Apel
109 |     obj_bnt = &apl;
110 |     obj_bnt->informasi();
111 |     obj_bnt->namaBuah();
112 |     obj_bnt->namaLatin();
113 |     obj_bnt->warna();
114 |
115 |
116 |     cout<<endl;
117 |
118 |     //menunjuk ke objek dari kelas Mangga

```

Gambar 2.4 tampilan program

```

116         cout<<endl;
117
118         //menunjuk ke objek dari kelas Mangga
119         obj_bnt = &mng;
120         obj_bnt->informasi();
121         obj_bnt->namaBuah();
122         obj_bnt->namaLatin();
123         obj_bnt->warna();
124     /*
125     cout<<"\tby"<<endl;
126     cout<<"\t  :"<<endl;
127     cout<<"\t    sugeng wahyu nugroho"<<endl;
128     */
129     _getche();
130
131     system("Pause");
132     return 0;
133 }

```

Gambar 2.5 tampilan program