LAPORAN HASIL PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN II

NAMA : REMEMBER SITOMPUL
NIM : 193020503022
KELAS : A
MODUL : III (POLIMORFISME)

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PALANGKA RAYA
2020

LAPORAN HASIL PRAKTIKUM

ALGORITMA DAN PEMROGRAMAN II

Nama : Remember sitompul

NIM : 193020503022

Kelas : A

Modul : III (Polimorfisme)

Komposisi	MAX	Nilai
BAB I Tujuan dan Landasan Teori	10	6
BAB II Pembahasan	60	45
BAB III Kesimpulan	20	10
Daftar Pustaka	5	3
Lampiran	5	4
Jumlah	100	

Penilai

Asisten Pratikum

Diana

BAB I

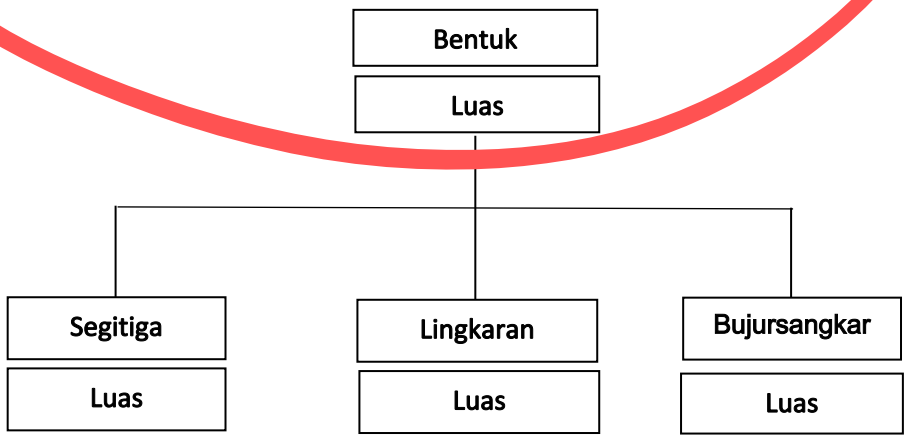
TUJUAN DAN LANDASAN TEORI

I. Tujuan

Setelah menyelesaikan modul ini, mahasiswa diharapkan mampu membuat polimorfisme.

II. Landasan Teori

Polimorfisme Memiliki arti "banyak bentuk", melakukan hal yang sama untuk berbagai data yang berbeda, mengirimkan pesan yang sama ke berbagai objek yang berbeda karena tiap objek memberi respons dengan cara yang berbeda. Berikut ini merupakan contoh polimorfisme.



Polimorfisme memiliki syarat-syarat sebagai berikut:

- Ada hirarki pewarisan
- Kelas dalam hirarki pewarisan barus memiliki fungsi virtual (virtual

method) dengan signature yang sama.

- Menggunakan pointer atau rujukan ke kelas induk. Pointer digunakan untuk memanggil fungsi virtual

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 batik, 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

BAB II

PEMBAHASAN

1.1 Program Pertama

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

//Kelas Dasar Ular (Kelas Abstrak)
class Binatang
{
protected:
    char nama_binatang[20];
    char suara_b[20];

public:
    void informasi()
    {
        cout<<"Informasi Binatang"<<endl;
    }

    //fungsi virtual murni
```

```
        virtual void namaBinatang() = 0;
        virtual void suara() = 0;
    };

    //kelas Kucing turunan dari kelas Binatang
    class Kucing : public Binatang
    {
    public:
        Kucing(char* nm, char *sr)
             {
            strcpy(nama_binatang, nm);
            strcpy(suara_b, sr);
        }

        void informasi()
        {
            cout<<"Informasi Kucing"<<endl;
        }

        void namaBinatang()
        {
            cout<<"Nama  : "<<nama_binatang<<endl;
        }

        void suara()
{
            cout<<"Suara : \""<<suara_b<<"\""<<endl;
        }
```

```
};

//kelas Bebek turunan dari kelas Binatang
class Bebek : public Binatang
{
public:
    Bebek(char* nm, char *sr)
    {
        strcpy(nama_binatang, nm);
        strcpy(suara_b, sr);
    }

    void informasi()
    {
        cout<<"Informasi Bebek"<<endl;
    }

    void namaBinatang()
    {
        cout<<"Nama  : "<<nama_binatang<<endl;
    }

    void suara()
    {
        cout<<"Suara : \""<<suara_b<<"\""<<endl;
    }
};
```

```
int main()
{
    //deklarasi objek
Binatang *obj_bnt;
    Kucing kcg("Kucing", "meow meow");
    Bebek bbk("Bebek", "kukuruyuk");



    cout<<"POLIMORFISME 2"<<endl;
    cout<<"----------"<<endl;

    //menunjuk ke objek dari kelas Kucing
    obj_bnt = &kcg;
    obj_bnt->informasi();
    obj_bnt->namaBinatang();
    obj_bnt->suara();

    cout<<endl;

    //menunjuk ke objek dari kelas Bebek
    obj_bnt = &bbk;
    obj_bnt->informasi();
    obj_bnt->namaBinatang();
    obj_bnt->suara();

    _getche();
    return 0;
}
```

Tabel 1.1 Program Polimorfisme class Hewan

Output:

```
POLIMORFISME 2
--------------
Informasi Binatang
Nama  : Kucing
Suara : "meow meow"

Informasi Binatang
Nama  : Bebek
Suara : "kukuruyuk"
```

Gambar 1.1 Hasil Output Program Polimorfisme class Hewan

Penjelasan :

Pada program class Binatang , coding program diawali dengan penulisan #include<iostream.h> . *include* sendiri adalah salah satu pengarah *prepoceccor directive* yang selalu dijalankan terlebih dahulu pada saat proses kompilasi terjadi.

Coding ***include*** tidak diakhiri dengan tanda **(;)**, karena bentuk tersebut bukanlah suatu bentuk pernyataan, tetapi merupakan prepocessor directive. Coding tersebut diakhiri dengan ekstensi **.h (file header)** yaitu file yang berisi sebagai deklarasi.

- #include<conio.h: pengarah compiler yang sering digunakan untuk menandai bahwa suatu file sudah diikutsertakan dalam kompilasi. Dan pada program tersebut, filenya adalah HEWAN.H
- #include <conio.h>: untuk mendefinisikan suatu pengenal / konstanta yang nantinya akan digantikan oleh praprosesor saat program dikompilasi.

Public adalah suatu tipe akses yang dapat diakses diluar class,sedangkan private diakses didalam class. Pada program diatas terdapat 3 kelas yaitu kelas Manusia, Pelajar, Pegawai. Dan

kelas utama pada program ini adalah kelas Hewan.

Pada kelas utama yaitu Manusia, menggunakan fungsi virtual, yaitu virtual void Informasi();. Fungsi virtual ini adalah pendefinisian ulang pada kelas-kelas turunannya. Jadi yang di ulang pada program ini adalah variabel Hello() yang bertipe void (merupakan nilai dari suatu variabel dan tidak bertipe data). Pada penulisan class Ular : public Hewan dan class Kucing : public Hewan , menandakan bahwa kelas Pelajar dan kelas Pegawai merupakan kelas turunan dari kelas Hewan Pada kedua kelas turunan tersebut menggunakan variabel **void Informasi();** dari kelas induk kelas hewan. Dan ketiga kelas ini menggunakan perubah akses (anggota kelas) public.

```
#include <iostream>
#include <conio.h>
using namespace std;
//Kelas Dasar Ular (Kelas Abstrak)
class Binatang
{
protected:
    char nama_binatang[20];
    char suara_b[20];
public:
    void informasi()
    {
        cout<<"Informasi Binatang"<<endl;
    }
    //fungsi virtual murni
    virtual void namaBinatang() = 0;
    virtual void suara() = 0;
};
```

```
//kelas Kucing turunan dari kelas Binatang
class Kucing : public Binatang
{
public:
    Kucing(char* nm, char *sr)
        {
        strcpy(nama_binatang, nm);
        strcpy(suara_b, sr);
    }
```

Penjelasan :

Pada bagian program yang merupakan program utama ini terdapat kekurangan dalam memberi masukan file lain. Jadi kita tambahkan fungsi #include "Hewan.h" untuk mengincludekan file Manusia.cpp yang kita buat sebelumnya. Dan fungsi #include <stdlib.h> adalah fungsi file header yang meliputi alokasi memori, kontrol proses, konversi dan lain-lain.

Int main () berfungsi sebagai penanda program utama. Suatu program pada C++ harus memiliki sebuah main. Main diikuti tanda () karena main merupakan sebuah fungsi. Pada penulisan Hewan.h; , tanda bintang (*) pada Manusia disebut pointer. Pointer (variabel penunjuk) adalah suatu variabel yang berisi alamat memori dari suatu variabel lain. Alamat ini merupakan lokasi dari obyek lain (biasanya variabel lain) di dalam memori. Dalam hal ini pointer menunjuk kelas Manusia, jadi pointer ini menghasilkan nilai yang berada pada kelas alamat yaitu variabel void Hello(). Pada program utama ini di buat variabel baru yaitu variabel pilihan yang bertipe integer (int piihan). Variabel ini digunakan untuk variabel inputan

untuk memilih pilihan pada penulisan berikut :

```
        void namaBinatang()
            {
                cout<<"Nama  : "<<nama_binatang<<endl;
            }
            void suara()
    {
               cout<<"Suara : \""<<suara_b<<"\""<<endl;
            }
```

Penjelasan :

void nama Binatang();. Fungsi virtual ini adalah pendefinisian ulang pada kelas-kelas turunannya. Jadi yang di ulang pada program ini adalah variabel Binatang() yang bertipe void (merupakan nilai dari suatu variabel dan tidak bertipe data).

```
        //kelas Bebek turunan dari kelas Binatang
        class Bebek : public Binatang
        {
        public: Hewan
            Bebek(char* nm, char *sr)
            {
                strcpy(nama_binatang, nm);
                strcpy(suara_b, sr);
            }
```

Penjelasan :

Pada penulisan class Bebek : public Kucing dan class ular : public

Hewan menandakan bahwa kelas Pelajar dan kelas Pegawai merupakan kelas turunan dari kelas Manusia. Pada kedua kelas turunan tersebut menggunakan variabel Fublic dari kelas induk kelas Manusia. Dan ketiga kelas ini menggunakan perubah akses (anggota kelas) public.

```
Binatang *obj_bnt;
        Kucing kcg("Kucing", "meow meow");
        Bebek bbk("Bebek", "kukuruyuk");
```

Penjelasan :

Kucing kcg (“Kucing” “ meow meow”);, Bebek bbk(“”Bebek”,”kukuruyuk); yang menggunakan untuk menampilkan masing-masing perintah yang ada pada script.

```
    cout<<"POLIMORFISME 2"<<endl;
        cout<<"----------"<<endl;

        //menunjuk ke objek dari kelas Kucing
        obj_bnt = &kcg;
        obj_bnt->informasi();
        obj_bnt->namaBinatang();
        obj_bnt->suara();

        cout<<endl;

        //menunjuk ke objek dari kelas Bebek
```

```
        obj_bnt = &bbk;
        obj_bnt->informasi();
        obj_bnt->namaBinatang();
        obj_bnt->suara();

        _getche();
        return 0;
}
```

Penjelasan :

Program di atas merupakan program dari analogi yang sebelumnya saya sampaikan di awal.
pada kelas dasar Binatang terdapat dua fungsi yang berbeda,

1. fungsi biasa;
2. fungsi virtual murni;

Sekarang perhatikan output program, ketika obj_bnt->informasi(); dijalankan hasilnya adalah string “Informasi Binatang”

Kenapa bukan “Informasi Kucing/Bebek” padahal pointer sudah menunjuk pada kelas turunan Kucing/Bebek? Hal ini dikarenakan kata “virtual” tidak ditambahkan pada fungsi informasi() yang ada pada kelas dasar Binatang (**sekedar mengulang apa yang jelaskan di awal)*. Dan jangan lupa juga bahwa fungsi informasi() dari kelas dasar Binatang telah diwariskan ke kelas turunan

BAB III

KESIMPULAN

Kesimpulan

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.

Pada polimorfisme menggunakan fungsi virtual, Fungsi virtual ini adalah pendefinisian ulang pada kelas-kelas turunannya.

Pointer (variabel penunjuk) adalah suatu variabel yang berisi alamat memori dari suatu variabel lain. Alamat ini merupakan lokasi dari obyek lain (biasanya variabel lain) di dalam memori.

Untuk melakukan pengulangan dapat menggunakan fungsi dari do…while switch Tanda -> digunakan untuk mengarahkan pointer kepada fungsi virtual yang telah di tentukan pada pengulangan struktur do…while switch.

DAFTAR FUSTAKA

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

BAB V

LAMPIRAN

1.2 Lampiran

```
POLIMORFISME 2
--------------
Informasi Binatang
Nama  : Kucing
Suara : "meow meow"

Informasi Binatang
Nama  : Bebek
Suara : "kukuruyuk"
```

Gambar Hasil Program Polimorfisme Pada Class Hewan