LAPORAN HASIL PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN II

NAMA : JORGI JACKO EXCEL
NIM : 193030503064
KELAS : A
MODUL : POLIMORFISME

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PALANGKA RAYA
2020

LAPORAN HASIL PRAKTIKUM

ALGORITMA DAN PEMROGRAMAN II

Nama : Jorgi Jacko Excel

NIM : 193030503064

Kelas : A

Modul : III (POLIMORFISME)

Komposisi	MAX	Nilai
BAB I Tujuan dan Landasan Teori	10	7
BAB II Pembahasan	60	50
BAB III Kesimpulan	20	13
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 LANDASAN 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.

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 digunakan untuk menyatakan suatu nama yang merujuk pada beberapa fungsi yang berbeda. Pada polimorfisme, rujukan dapat dilakukan pada berbagai tipe objek. Hal ini dilakukan karena setiap objek dimungkinkan memiliki instruksi yang berbeda. Dalam mengimplementasikan polimorfisme, perlu diperhatikan hal-hal sebagai berikut:

a. Method yang dipanggil harus melalui variabel dari super class.
b. Method yang dipanggil juga harus merupakan method yang ada pada super class.
c. Signature method harus sama baik yang ada pada super class maupun di subclass.
d. Method access attribute pada subclass tidak boleh lebih terbatas daripada yang ada pada super class.

a. Polimorfisme pada pewarisan tunggal

Suatu metode dapat dideklarasikan secara virtual dengan cara menambahkan virtual pada deklarasi fungsi. Jika suatu metode dideklarasikan secara virtual dalam kelas dasar, maka keadaan virtual tetap akan berlaku untuk fungsi yang sama di seluruh kelas turunan meskipun kata virtual tidak disebutkan.

b. Polimorfisme pada pewarisan jamak

Polimorfisme juga dapat diterapkan pada pewarisan jamak, tidak ada bedanya dengan [ewarisan tunggal, hanya saja kata virtual harus disebutkan pada semua metode pada kelas dasar yang dirancang dapat mengadakan ikatan dinamis.

c. Polimorfisme pada pewarisan jamak virtual

Polimorfisme pada pewarisan jamak virtual tidak hanya berbeda. Misalnya pada eklas hewan dan Pegasus, untuk menyederhanakan program hanya diperhatikan metode bergerak() pada kelas hewan. Metode bergerak pada kelas hewan berbeda dengan metode bergerak pada kelas kuda ataupun Pegasus.

BAB II

PEMBAHASAN

1. Program polimorfisme buah

Program ini terdiri dari tiga bagian program yaitu kelas header, kelas implementasi dan program utama dimana lembar-lembar pengkodingan program ini terpisah, tidak dalam satu lembar kerja. Berikut adalah bagian-bagian program pertama dan pembahasannya.

a. Buah.h

Pertama hal yang dilakukan adalah menuliskan file header dari program ini. File header adalah sebuah file yang digunakan untuk mendefinisikan beberapa file macro, fungsi, variabel dan konstan. File header nantinya akan mengandung beberapa fungsi atau perintah yang akan digunakan dalam program. Di bawah ini adalah file header dari bagian pertama ini.

```
#ifndef BUAH_H
#define BUAH_H
```

Diatas merupakan sebuah directive#ifndef dan directive#define yang berfungsi untuk mendefinisikan sebuah makro identifier ke dalam program, dimana makro identifier yang dimaksud disini adalah BUAH.

Berikutnya adalah mendeklarasikan kelas-kelas yang ada dalam program bagian pertama ini.

Pertama adalah kelas **Buah**

```
class Buah {
        public:
        virtual void Rasa();
};
```

Kelas Buah ini adalah kelas induk. Dari pendeklarasian kelas di atas, hak akses yang diberikan adalah *Public*, yang artinya dapat diakses secara umum kepada kelas-kelas turunannya. Dalam atribut public berisi sebuah

virtual yang akan menampilkan Rasa yang bertipe void. Fungsi virtual ini adalah untuk pendefinisian pada class class turunannya. Program di atas adalah memerintahkan untuk menentukan Rasa sebagai outputnya.

Kedua adalah kelas **Jeruk**

```
class Jeruk : public Buah {
        public:
        void Rasa();
};
```

Kelas Jeruk ini merupakan kelas turunan dari kelas Manusia. Hal tersebut ditandai dengan **class Jeruk : public Buah**. Program di atas sama dengan kelas induknya yang memiliki hak akses *Public* dan memerintahkan untuk menentuka Rasa pada layar output.

Ketiga adalah kelas **Mangga**

```
class Mangga : public Buah {
        public:
        void Rasa();
};
```

Kelas Mangga ini juga merupakan kelas turunan dari kelas Manusia, yang ditandai dengan coding **class Mangga : public Buah**. Program di atas sama dengan kelas induknya yang memiliki hak akses *Public* dan memerintahkan untuk menentukan kata Rasa pada layar output.

Ketiga adalah kelas **Mangga**

```
class Rambutan : public Buah {
        public:
        void Rasa();
};
#endif
```

Kelas Rambutan ini juga merupakan kelas turunan dari kelas Manusia, yang ditandai dengan coding **class Rambutan : public Buah**. Program di atas sama dengan kelas induknya yang memiliki hak akses *Public* dan memerintahkan untuk menentukan Rasa pada layar output. **#endif** merupakan akhir dari program bagian pertama ini.

b. Buah.cpp

Selanjutnya adalah masuk pada bagian kedua program yaitu kelas implementasi. Sama seperti pada bagian pertama, hal pertama yang dilakukan adalah menuliskan file header terlebih dahulu. Berikut adalah file header bagian kedua ini :

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

Pernyataan yang diawali dengan tanda # adalah pernyataan untuk menyatakan preprocessor, pernyataan tersebut bukan untuk di eksekusi. Pernyataan **#include "Buah.h"** berarti menginputkan file header Buah.h ke dalam Buah.cpp. fungsi **include <iostream>** adalah untuk menjalankan perintah *cout*, karena *cout* merupakan bagian dari iostream.h. Fungsi ***using namespace std*** untuk mengijinkan mengakses pustaka standar pada C++.

Berikutnya adalah masuk pada bagian prosedur-prosedur pada bagian kedua ini.

Prosedur pertama :

```
void Buah::Rasa() {
        cout<<"====================="<<endl;
        cout<<"Macam Macam Rasa Buah adalah:"<<endl;
        cout<<"Manis, Masam, dan Hambar"<< endl;
}
```

Void dalam bahasa C++ juga bisa disebut sebagai prosedur. Perintah **cout** berfungsi untuk melakukan output ke peralatan standar. Adapun operator yang digunakan adalah operator **<<**. Perintah **endl** berguna menampilkan baris baru dan membuang stream. Prosedur di atas memberi intruksi bahwa cetak kata "*Macam Macam Rasa Buah*" dan "*Manis, Masam, dan Hambar"* untuk kelas Buah yang akan mengembalikan nilai ke variabel **Rasa** yang telah dideklarasikan pada bagian pertama.

Prosedur kedua:

```
void Jeruk::Rasa() {
        cout<<"====================="<<endl;
        cout<<"Rasa Jeruk adalah:"<<endl;
        cout<<"Manis dan Masam"<< endl;
}
```

Pengertian dari prosedur kedua ini sama dengan prosedur pertama tadi. Namun prosedur ini memberi intruksi bahwa cetak kata "*Rasa Jeruk Adalah*" dan "*Manis dan Masam"* untuk kelas Jeruk yang akan mengembalikan nilai ke variabel **Rasa** yang telah dideklarasikan pada bagian pertama.

Prosedur ketiga:

```
void Mangga::Rasa() {
        cout<<"====================="<<endl;
        cout<<"Rasa Mangga adalah:"<<endl;
        cout<<"Manis dan Masam"<<endl;
}
```

Pengertian dari prosedur ketiga ini sama dengan prosedur pertama dan kedua tadi. Namun prosedur ini memberi intruksi bahwa cetak kata "*Rasa Mangga Adalah*" dan "*Manis dan Masam"* untuk kelas Mangga yang akan mengembalikan nilai ke variabel **Rasa** yang telah dideklarasikan pada bagian pertama.

Prosedur keempat:

```
void Rambutan::Rasa(){
        cout<<"====================="<<endl;
        cout<<"Rasa Rambutan adalah:"<<endl;
        cout<<"Manis"<<endl;
}
```

Pengertian dari prosedur keempat ini sama dengan prosedur pertama dan ketiga tadi. Namun prosedur ini memberi intruksi bahwa cetak kata "*Rasa Mangga Rambutan*" dan "*Manis"* untuk kelas Rambutan yang akan mengembalikan nilai ke variabel **Rasa** yang telah dideklarasikan pada bagian pertama.

c. Main.cpp

Selanjutnya adalah masuk pada bagian ketiga program yaitu program utama. Sama seperti pada bagian pertama, hal pertama yang dilakukan adalah menuliskan file header terlebih dahulu. Berikut adalah file header bagian ketiga ini :

```
#include "Buah.h"
#include <iostream>
#include <stdlib.h>
using namespace std;
```

Pernyataan yang diawali dengan tanda (#) merupakan pernyataan untuk menyertakan preprocessor, pernyataan ini tidak untuk dieksekusi. Fungsi **#include "Buah.h"** berarti memasukkan file Buah.h ke dalam Main.cpp. Fungsi **#include <iostream>,** dalam fungsi ini terdapat beberapa fungsi standar, yaitu perintah cout yang merupakan bagian dari file header iostream. Dan fungsi **#include <stdblib.h>**, adalah fungsi yang meliputi alokasi memori, control process, dll. Fungsi **using namespace std** untuk mengijinkan mengakses pustaka standar pada C++.

Berikutnya masuk program utama dari program pertama ini. Disinilah semua intruksi-intruksi yang telah dituliskan sebelumnya akan dipanggil dalam program. Berikut adalah coding dari program utama ini :

```
int main() {
Buah* b;
int pilihan;
do {
        cout<<"Pilih Salah Satu Rasa"<<endl;
        cout<<"1. Macam Macam Rasa Buah, 2. Jeruk, 3.
Mangga, 4.Rambutan:  "; cin >> pilihan;
} while ( pilihan < 1 || pilihan > 4 );
switch ( pilihan ) {
case 1: b = new Buah; break;
case 2: b = new Jeruk; break;
case 3: b = new Mangga; break;
case 4: b = new Rambutan; break;
}
b->Rasa();
delete b;
return 0;
}
```

main( ) adalah sebuah fungsi utama di dalam sebuah program. Fungsi inilah yang akan dipanggil pertama kali pada saat eksekusi program. Apabila ada fungsi lain yang dibuat, maka fungsi tersebut akan dijankan ketika dipanggil di fungsi utama.

Pada implementasi class, dilakukan pemanggilan file Buah.h dan pendeklarasian fungsi Rasa dari masing-masing kelas. Pada kelas Jeruk, Mangga, dan Rambutan dilakukan override terhadap fungsi Rasa. Hal yang dilakukan pertama pada program utama adalah memanggil file Buah.cpp dan mendeklarasikan pointer b yang akan menunjuk kelas Buah. Pointer

yang menunjuk kelas Buah akan menghasilkan nilai yang berada pada kelas alamat yaitu variabel *void Rasa ( )*. Pada program utama ini juga dibuat variabel pilihan yang bertipe integer. Selanjutnya pada pemilihan kita menggunakan struktur do..while. dan pada pengulangan dilakukan perulangan pada fungsi cout, dan fungsi cin untuk menggunakan variabel pilihan sebagai variabel inputan. Pada **struktur while**, while (pilihan < 1 || pilihan > 4); , pengulangan akan terus dilakukan apabila inputan yang dimasukkan kurang dari 1 atau lebih dari 4. Pada struktur switch setelah pilihan ditentukan, maka akan terpilih sesuai yang terstruktur di bagian switch. Dan pointer **b** merujuk kepada fungsi Rasa() yang ada pada pilihan yang kita tentukan. Menjalankan intruksi dari inputan yang akan diberikan (selama inputan pilihan bernilai 1-4), jika pilihan 1 maka akan menampilkan fungsi virtual Rasa pada kelas Buah, jika pilihan 2 maka akan menampilkan fungsi Rasa pada kelas Jeruk, jika pilihan 3 maka akan menampilkan fungsi Rasa pada kelas Mangga, dan jika pilihan 4 maka akan menampilkan fungsi Rasa pada kelas Rambutan.

new digunakan untuk mengalokasikan memori pada ruang yang masih kosong. Intruksi ini akan diikuti oleh tipe data yang akan dialokasikan sehingga kompiler akan mengetahui seberapa besar ruang memori yang dibutuhkan untuk proses pengalokasian tersebut. Pada saat mengalokasikan memori, alamat memori yang dialokasikan tersebut tentu akan disimpan ke pointer, sehingga sebelumnya harus mendeklarasikan pointer terlebih dahulu. **break** berfungsi untuk menghentikan proses pengulangan dan program akan langsung meloncat ke statemen yang berada di bawah blok pengulangan yang bersangkutan. **delete** berfungsi untuk mendealokasikan kembali memori yang sudah tidak digunakan lagi. Fungsi **return 0** pada konsep di atas adalah fungsi main() ini dapat mengembalikan nilai 0 ke system operasi yang menandakan bahwa program tersebut berjalan dengan baik tanpa adanya kesalahan.

Buah.cpp dan Main.cpp saling berhubungan, Main.cpp juga berhubungan dengan Buah.h. Jadi, apabila dalam Main.cpp tidak ada salah

satu pun Buah.cpp dan Buah.h tidak dipanggil, maka program tersebut akan terjadi error atau terjadi kesalahan. Begitu pula apabila terdapat error pada salah satu program, maka program yang selanjutnya juga akan mengalami error.

BAB III

KESIMPULAN

Polymorphism adalah kemampuan untuk menggunakan operator atau fungsi dalam berbagai cara. Polimorfisme memberikan arti yang berbeda atau fungsinya kepada operator atau fungsi. Poly, merujuk ke banyak, menandakan banyak kegunaan dari operator dan fungsi ini. Fungsi tunggal penggunaan atau operator berfungsi dalam banyak cara bisa disebut polimorfisme. Polimorfisme mengacu pada kode, operasi atau benda yang berperilaku berbeda dalam konteks yang berbeda.

Polimorfisme adalah fitur yang kuat dari bahasa pemrograman berorientasi obyek C + +.Sebuah operator + berperilaku berbeda dalam konteks yang berbeda, seperti integer, float atau string yang mengacu konsep polimorfisme. Konsep di atas mengarah ke operator overloading. Konsep overloading juga merupakan cabang dari polimorfisme. Ketika keluar operator atau fungsi yang beroperasi pada tipe data baru itu kelebihan beban. Polimorfisme fitur ini mengarah pada konsep metode virtual.

DAFTAR PUSTAKA

Dosen Teknik Informatika. *Modul Praktikum Algoritma Pemrograman II*. Palangkaraya: Jurusan Teknik Informatika, 2020. Print.

Rizki, Aditya. 2011. *Tutorial pemrograman berorientasi objek dengan Polimorfisme Studi Kasus.* http://www.adityarizki.net/2011/04/tutorial-pemrograman-berorientasi-obyek-dengan-c-polimorfisme-studi-kasus/ Diakses pada 21 April 2020

Rizky, Fikrii. 2012. *Polimorfisme dan Inheritance*.http://fikriirizky.blogspot.com /2012/12/polimorfisme-dan-inheritance-pada.html Diakses pada 21 April 2020

LAMPIRAN

Source code dalam c++

buah.cpp | buah.h | main.cpp

```
1  #ifndef BUAH_H
2  #define BUAH_H
3  class Buah {
4  public:
5  virtual void Rasa();
6  };
7  class Jeruk : public Buah {
8  public:
9  void Rasa();
10 };
11 class Mangga : public Buah {
12 public:
13 void Rasa();
14 };
15 class Rambutan : public Buah {
16 public:
17 void Rasa();
18 };
19 #endif
```

Gambar 1 *buah.h*

buah.cpp | buah.h | main.cpp

```
1  #include "Buah.h"
2  #include <iostream>
3  using namespace std;
4
5  void Buah::Rasa() {
6      cout<<"====================="<<endl;
7      cout<<"Macam Macam Rasa adalah:"<<endl;
8      cout<<"Manis, Masam, dan Hambar"<< endl;
9  }
10 void Jeruk::Rasa() {
11     cout<<"====================="<<endl;
12     cout<<"Rasa Jeruk adalah:"<<endl;
13     cout<<"Manis dan Masam"<< endl;
14 }
15 void Mangga::Rasa() {
16     cout<<"====================="<<endl;
17     cout<<"Rasa Mangga adalah:"<<endl;
18     cout<<"Manis dan Masam"<<endl;
19 }
20 void Rambutan::Rasa(){
21     cout<<"====================="<<endl;
22     cout<<"Rasa Rambutan adalah:"<<endl;
23     cout<<"Manis"<<endl;
24 }
25
```

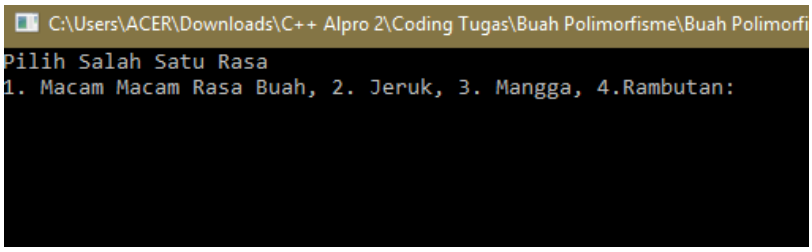
Gambar 2 *Buah.cpp*

buah.cpp | buah.h | main.cpp

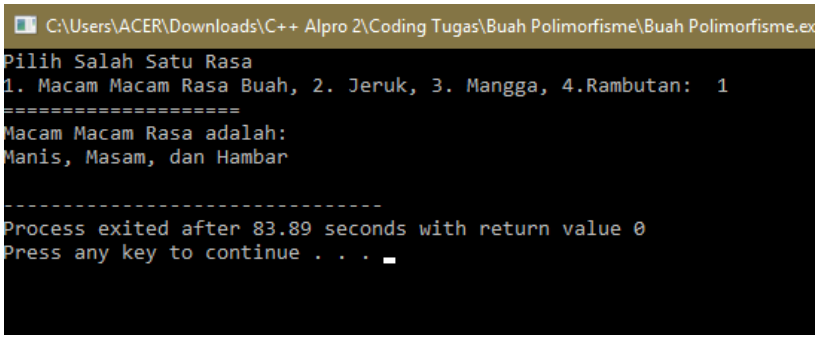
```cpp
#include "Buah.h"
#include <iostream>
#include <stdlib.h>
using namespace std;

int main() {
Buah* b;
int pilihan;
do {
    cout<<"Pilih Salah Satu Rasa"<<endl;
    cout<<"1. Macam Macam Rasa Buah, 2. Jeruk, 3. Mangga, 4.Rambutan:  "; cin >> pilihan;
} while ( pilihan < 1 || pilihan > 4 );
switch ( pilihan ) {
case 1: b = new Buah; break;
case 2: b = new Jeruk; break;
case 3: b = new Mangga; break;
case 4: b = new Rambutan; break;
}
b->Rasa();
delete b;
return 0;
}
```

Gambar 3 *main.cpp*

Output program



Gambar 4 *tampilan menu*



Gambar 5 *tampilan hasil hari menu pertama*