

**LAPORAN HASIL PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN II**



NAMA : PRAYOGA BETRIO L
NIM : 193030503060
KELAS : A
MODUL : III (FOLIMORFISME)

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PALANGKA RAYA
2020

**LAPORAN HASIL PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN II**



Nama : PRAYOGA BETRIO L
NIM : 193030503060
Kelas : A
Modul : III (FOLIMORFISME)

| Komposisi | MAX | Nilai |
|---------------------------------|-----|-------|
| BAB I Tujuan dan Landasan Teori | 10 | 8 |
| BAB II Pembahasan | 60 | 48 |
| BAB III Kesimpulan | 20 | 13 |
| Daftar Pustaka | 5 | 5 |
| Lampiran | 5 | 5 |
| Jumlah | 100 | |

Penilai
Asisten Praktikum

Diana

BAB I

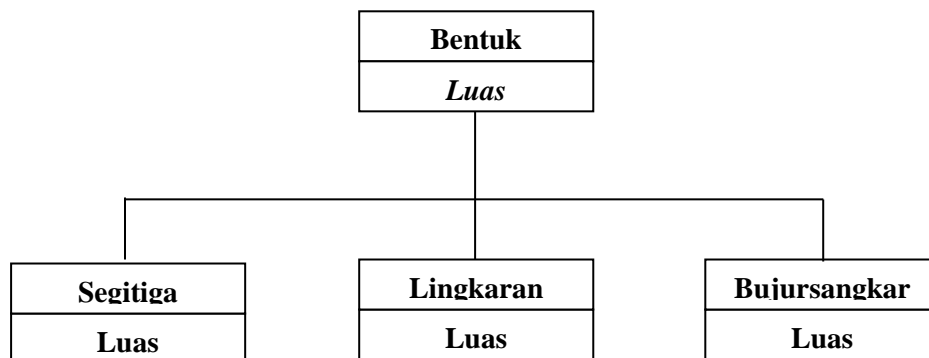
TUJUAN DAN LANDASAN TEORI

I. TUJUAN

Setelah menyelesaikan modul ini, mahasiswa diharapkan mampu membuat polimorfisme.

I. 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 harus 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 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

Polimorfisme adalah kemampuan suatu objek untuk mengungkap banyak hal melalui suatu cara yang sama. Sebagai contoh, terdapat kelas A yang diturunkan menjadi kelas B, C, dan D. Dengan konsep polimorfisme, kita dapat menjalankan method-method yang terdapat pada kelas B, C, dan D hanya dari objek yang diinstansiasi dengan kelas A. Polimorfisme sering dinamakan dengan dynamic binding, late binding, maupun runtime binding.

Polimorfisme secara leksikal bahasa berarti ‘banyak bentuk’. Dalam konsep OOP, polimorfisme adalah pendefinisian suatu perilaku-perilaku objek yang memiliki nama yang sama tetapi mengolah masukan yang berbeda. Masukan yang berbeda menyebabkan detail implementasi perilakunya juga berbeda.

Sebagai contoh, objek orang memiliki perilaku ‘makan’. Masukan atau input perilaku ‘makan’ untuk setiap orang berbeda-beda tergantung kebiasaan (dan tentunya juga wilayah). Orang Indonesia biasa makan nasi, sedangkan orang Eropa biasa makan roti. Polimorfisme mengizinkan pendefinisian perilaku ‘makan’ ini secara berulang (dengan nama yang sama) asalkan input yang diberikan tipenya berbeda-beda. Polimorfisme secara bahasa dapat diartikan memiliki banyak bentuk. Konsep ini terdapat dalam bahasa pemrograman seperti konstruktor yang memiliki beberapa bentuk. Selain konstruktor, konsep ini juga berlaku bagi metode. Metode atau konstruktor dapat memiliki banyak bentuk dalam arti memiliki nama yang sama namun dengan argumen yang berbeda atau dengan return type yang berbeda. Contoh polimorfisme untuk konstruktor maupun untuk metode dapat Anda lihat pada Listing 1. Di sana terdapat konstruktor-konstruktor dengan nama sama namun dengan argumen yang mengandung parameter-parameter yang berbeda. Untuk contoh polimorfisme untuk metode ditunjukkan bahwa terdapat metode dengan nama sama namun memiliki argumen dan return type yang berbeda.

Kegunaan dari polimorfisme adalah agar kita dapat mendefinisikan beberapa konstruktor atau metode dengan karakteristik yang berbeda-beda agar nantinya dapat digunakan untuk kasus-kasus yang berbeda. Misalnya kita ingin menciptakan instans dari kelas `KelasKita` pada Listing 1 tanpa memberikan nilai apapun, namun terkadang kita ingin memberikan sebuah nilai sebagai parameter untuk digunakan oleh instans dari kelas tersebut, maka kita dapat membuat kelas seperti `KelasKita` tersebut. Begitu juga halnya dengan metode, sehingga kita dapat membuat metode-metode yang memiliki karakteristik yang khusus. Polimorfisme sebenarnya dapat dihilangkan dengan mendefinisikan sebuah konstruktor atau metode yang dapat menangani semua kasus yang mungkin. Namun hal ini akan menyebabkan program Anda lebih rumit dan sulit dimengerti. Sedangkan polimorfisme yang membantu Anda untuk membuat program yang lebih baik dan mudah juga membawa konsekuensi yaitu proses kompilasi yang lebih rumit, dan penurunan kecepatan eksekusi kelas. Namun hal ini tidak perlu menjadi perhatian Anda kecuali Anda memang ingin mendalami proses kompilasi Java. Polimorfisme 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. Contohnya, bila sebuah burung menerima pesan "gerak cepat", dia akan menggerakkan sayapnya dan terbang. Bila seekor singa menerima pesan yang sama, dia akan menggerakkan kakinya dan berlari. Keduanya menjawab sebuah pesan yang sama, namun yang sesuai dengan kemampuan hewan tersebut. Ini disebut polimorfisme karena sebuah variabel tunggal dalam program dapat memegang berbagai jenis objek yang berbeda selagi program berjalan, dan teks program yang sama dapat memanggil beberapa metode yang berbeda di saat yang berbeda dalam pemanggilan yang sama. Hal ini berlawanan dengan bahasa fungsional yang mencapai polimorfisme melalui penggunaan fungsi kelas-pertama. Polymorphism berasal dari bahasa Yunani yang berarti banyak bentuk. Dalam PBO, konsep ini memungkinkan digunakannya suatu interface yang sama untuk memerintah objek agar melakukan aksi atau tindakan yang mungkin

secara prinsip sama namun secara proses berbeda. Dalam konsep yang lebih umum sering kali polymorphism disebut dalam istilah satu interface banyak aksi. Contoh yang konkrit dalam dunia nyata yaitu mobil. Mobil yang ada dipasaran terdiri atas berbagai tipe dan berbagai merk, namun 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.

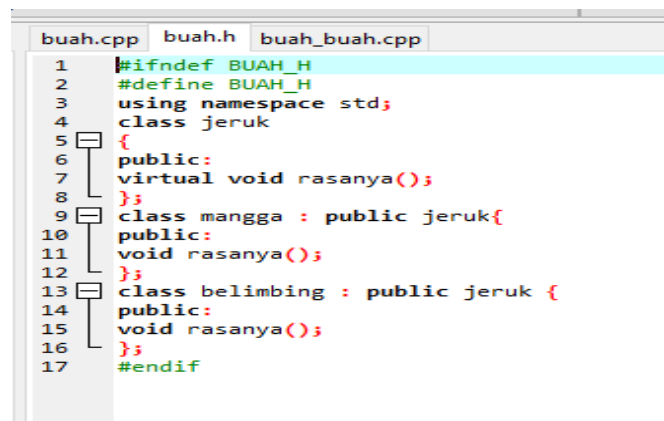
Harus diperhatikan disini bahwa interface yang sama tidak berarti cara kerjanya juga sama. Misal pedal gas, jika ditekan maka kecepatan mobil akan meningkat, tapi bagaiman proses peningkatan kecepatan ini dapat berbeda-beda untuk setiap jenis mobil. Polimorfisme adalah kemampuan dari sebuah object untuk membolehkan mengambil beberapa bentuk yang berbeda. Secara harfiah, “poli” berarti banyak sementara “morph” berarti bentuk.

BAB II

PEMBAHASAN

2.1 Program tugas :

Buah.h



```
1  #ifndef BUAH_H
2  #define BUAH_H
3  using namespace std;
4  class jeruk
5  {
6  public:
7      virtual void rasanya();
8  };
9  class mangga : public jeruk{
10 public:
11     void rasanya();
12 };
13 class belimbing : public jeruk {
14 public:
15     void rasanya();
16 };
17 #endif
```

Gambar 2.1.1

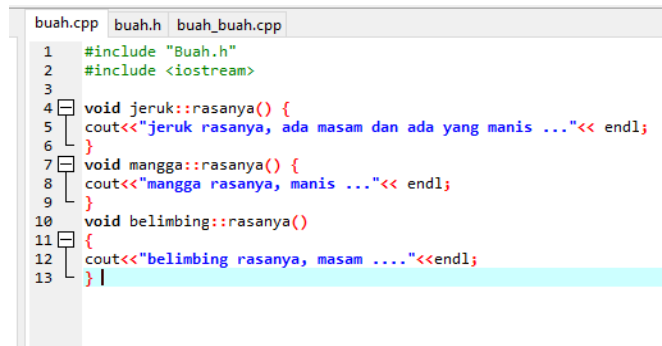
- Langkah pertama, kita harus membuat file headernya terlebih dahulu, file header ini diberi judul **Buah.h**.

Pada file header ini digunakan dua buah preprocessor yaitu *#ifndef* yang digunakan pada beberapa porsi variasi compile program secara selektif dan *#define* yang digunakan untuk melaksanakan substitusi makro dari satu lembar teks ke lembar teks yang lain melalui suatu file dimana teks tersebut digunakan. Preprocessor merupakan instruksi pada C++ yang digunakan untuk memberikan suatu perintah pada compiler. Setelah memasukkan preprocessornya, kita membuat class jeruk disini, yang berfungsi sebagai enkapsulasi yang menggunakan mode akses public. Dalam C++, mode akses public ini sendiri merupakan mode akses yang dimana variabel dari class itu bisa diakses oleh class itu sendiri maupun oleh class-class turunannya, juga bisa langsung dieksekusi oleh program.

Class *jeruk* dalam mode akses publicnya memiliki variabel *rasanya* yang tipe datanya virtual void. Pada class *mangga* dan *belimbing*, masing-masing mode akses publicnya memiliki variabel *rasanya* yang tipe datanya void. Void

merupakan tipe data yang digunakan untuk tipe suatu fungsi yang tidak mengembalikan nilai keluarannya. Sedangkan virtual void adalah sebuah fungsi yang dideklarasikan sebagai virtual dalam kelas induk dan didefinisikan kembali di dalam satu kelas turunan atau lebih. #endif digunakan untuk menandai akhir sebuah blok program.

Buah.cpp



```
1  #include "Buah.h"
2  #include <iostream>
3
4  void jeruk::rasanya() {
5      cout<<"jeruk rasanya, ada masam dan ada yang manis ..."<< endl;
6  }
7  void mangga::rasanya() {
8      cout<<"mangga rasanya, manis ..."<< endl;
9  }
10 void belimbing::rasanya()
11 {
12     cout<<"belimbing rasanya, masam ...."<<endl;
13 }
```

Gambar 2.1.2

- Langkah yang selanjutnya adalah membuat source file yang diberi judul **Buah.cpp**.

Pada source file Buah.cpp ini, menggunakan header #include "Buah.h" dan #include <iostream.h>. Header include "Buah.h" merupakan header yang telah dibuat pada langkah yang pertama tadi. Isi dari file ini merupakan hasil output dari masing-masing class pada program jika program dijalankan. Sedangkan header #include <iostream.h> diperlukan pada program yang melibatkan objek cout dan cin. Pada program ini juga digunakan **cout** yang berfungsi untuk menampilkan data ke dalam output (cetak pada layar). Tanda '<<' merupakan operator yang fungsinya sebagai penghubung antara stream dengan kalimat inputan. **endl** merupakan manipulator yang digunakan untuk menyisipkan karakter mengatur pindah baris pada program. Tanda ';' artinya menyatakan bahwa akhir dari sebuah pernyataan. Tanda '{' dan '}' berfungsi untuk menandakan awal dan akhir sebuah definisi fungsi dan program.

Buah_buah.cpp


```

buah.cpp  buah.h  buah_buah.cpp
1  #include "Buah.h"
2  #include <iostream>
3  #include <stdlib.h>
4  int main()
5  {
6      jeruk* b;
7      int pilihan;
8      do {
9          cout<<"1: jeruk, 2: mangga, 3: belimbing >> ";
10         cin >> pilihan;
11         } while ( pilihan < 1 || pilihan > 3 );
12
13     switch ( pilihan ) {
14     case 1: b = new jeruk; break;
15     case 2: b = new mangga; break;
16     case 3: b = new belimbing; break;
17     }
18     b->rasanya();
19     delete b;
20
21     return 0;

```

Gambar 2.1.3

- Langkah yang terakhir adalah pembuatan source file untuk bagian utama program yang diberi judul **Buah_buah.cpp**.

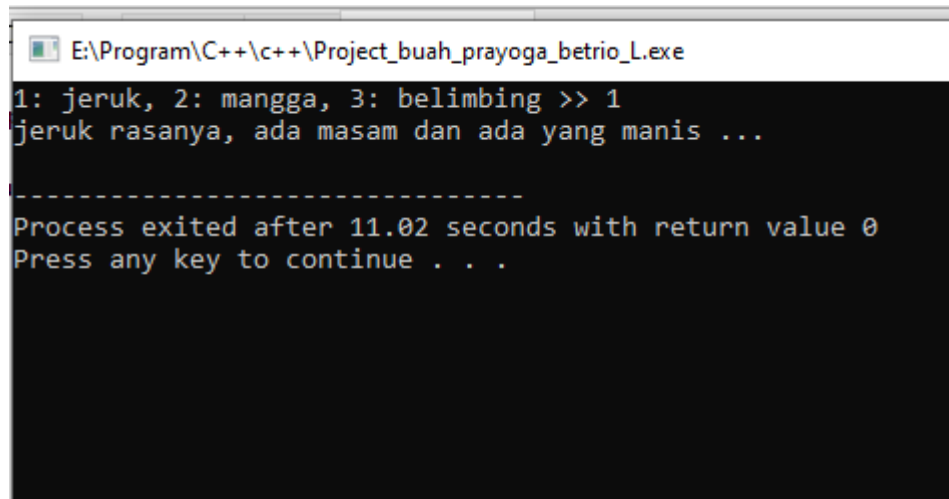
Pada file **Buah_buah.cpp** ini digunakan header `#include "Buah.h"` adalah header yang telah dibuat pada langkah awal tadi, `#include <iostream.h>` diperlukan dalam program yang melibatkan `cout` dan `cin`, serta `#include <stdlib.h>` yang digunakan untuk menjabarkan beberapa fungsi umum dan macro termasuk manajemen memori dinamis, menjalin komunikasi dengan perangkat sekitar, membuat bilangan secara random, aritmatika bilangan integer, pencarian, pengurutan dan pengkonversian.

int main() { menandakan awalan dari bagian utama program dan tanda }menandakan berakhirnya program. Pada program ini juga dilakukan fungsi pemilihan. Pesan pada masing-masing class, baik itu class jeruk, class mangga, maupun class belimbing akan ditampilkan jika kita telah melakukan pemilihan saat program dijalankan.

Pada program ini digunakan fungsi `switch – case`, yaitu pernyataan yang dirancang khusus untuk menangani pengambilan keputusan yang melibatkan sejumlah atau banyak alternatif penyelesaian. Mirip dengan penggunaan `case – of` pada Pascal, fungsi ini memungkinkan user untuk memilih salah satu dari pilihan yang ditetapkan dari berbagai ekspresi. Pemilihan berbagai kemungkinan nilai `switch` dilakukan satu demi satu berdasarkan nilai `case`. Jika

nilai dalam ekspresi switch tidak ada yang sesuai dengan nilai-nilai case, maka pilihan akan secara otomatis dialihkan ke default. **Return** digunakan karena pada main program menggunakan tipe data integer. Tanda '}' menandakan berakhirnya program.

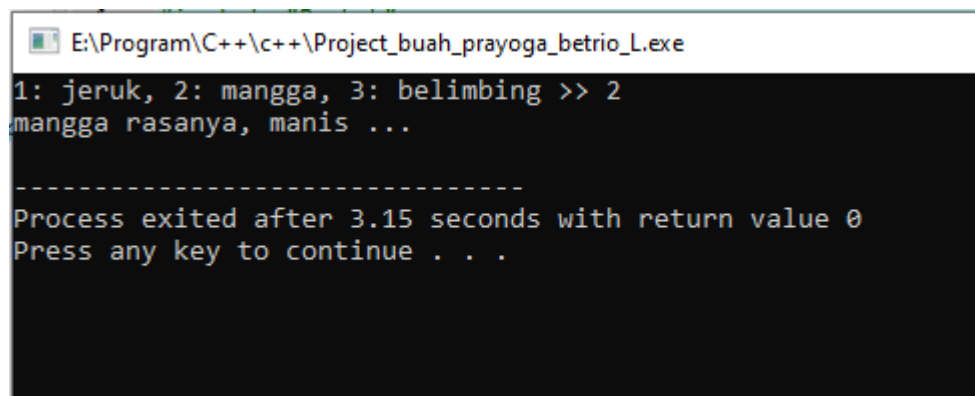
hasil outputnya :



```
E:\Program\C++\c++\Project_buah_prayoga_betrio_L.exe
1: jeruk, 2: mangga, 3: belimbing >> 1
jeruk rasanya, ada masam dan ada yang manis ...

-----
Process exited after 11.02 seconds with return value 0
Press any key to continue . . .
```

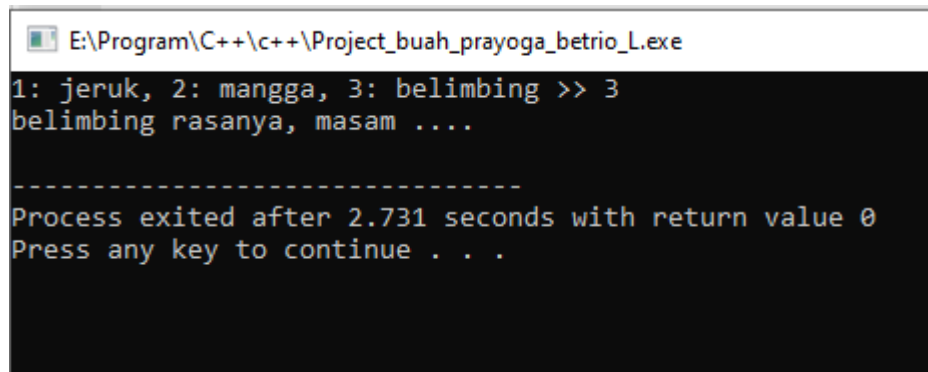
Gambar 2.1.4



```
E:\Program\C++\c++\Project_buah_prayoga_betrio_L.exe
1: jeruk, 2: mangga, 3: belimbing >> 2
mangga rasanya, manis ...

-----
Process exited after 3.15 seconds with return value 0
Press any key to continue . . .
```

Gambar 2.1.5



```
E:\Program\C++\c++\Project_buah_prayoga_betrio_L.exe
1: jeruk, 2: mangga, 3: belimbing >> 3
belimbing rasanya, masam ....

-----
Process exited after 2.731 seconds with return value 0
Press any key to continue . . .
```

Gambar 2.1.6

Program tugas ini merupakan program yang memiliki tiga pilihan pada menu outputnya, yaitu jeruk, mangga dan belimbing yang jika dipilih akan menampilkan pesan yang berbeda. Jika user memilih menu jeruk, maka pesan yang akan ditampilkan adalah **“jeruk rasanya, ada masam dan ada yang manis ...”**. Jika user memilih menu mangga, maka pesan yang akan ditampilkan adalah **“mangga rasanya, manis ...”**. Selanjutnya, jika user memilih menu belimbing, maka program akan menampilkan pesan **“belimbing rasanya, masam ...”**. Dalam pembuatannya, program ini memerlukan satu buah header yang diberi judul Buah.h dan dua buah source file yang diberi judul Buah.cpp dan buah_buah.cpp. Ketiga bagian program ini dibuat terpisah (tidak langsung digabung menjadi satu file) dan disimpan pada folder yang sama. Untuk kemudian ketiga file program ini akan dibuka dan diexecute secara bersamaan.

BAB III

KESIMPULAN

I. Kesimpulan

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.

Kegunaan dari polimorfisme adalah agar kita dapat mendefinisikan beberapa konstruktor atau metode dengan karakteristik yang berbeda-beda agar nantinya dapat digunakan untuk kasus-kasus yang berbeda.

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

Polimorfisme mengacu pada kemampuan untuk memanggil fungsi-fungsi yang berbeda dengan menggunakan hanya satu jenis fungsi panggil. Dalam C++ hal ini dapat dicapai dengan menggunakan fungsi virtual. Fungsi virtual adalah sebuah fungsi yang dideklarasikan sebagai virtual dalam kelas induk dan didefinisikan kembali di dalam satu kelas turunan atau lebih. Untuk mendeklarasikan sebuah fungsi virtual dalam kelas induk, kita cukup menambahkan kata kunci virtual.

DAFTAR PUSTAKA

Modul Praktikum Algoritma dan Pemrograman II, Jurusan Teknik Informatika
Universitas Palangkaraya (UPR). Angkatan 2020.

Helentrisna. 2009. Polimorfisme. Diakses 20 April 2020 jam 08.00 WIB.

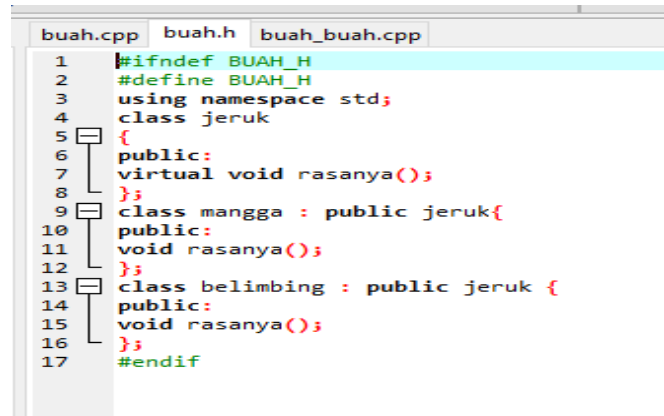
<http://blogspot.com/2009/12/pilimorfisme.html>

Iqbal.Taufioourrachman.Published on Mar 25, 2016. C19 Mutasi dan
Polimorfisme. Diakses 21 april 2020 jam 20.54 WIB.

<https://www.slideshare.net/itaufiqurrachman/c19-mutasi-dan-polimorfisme>

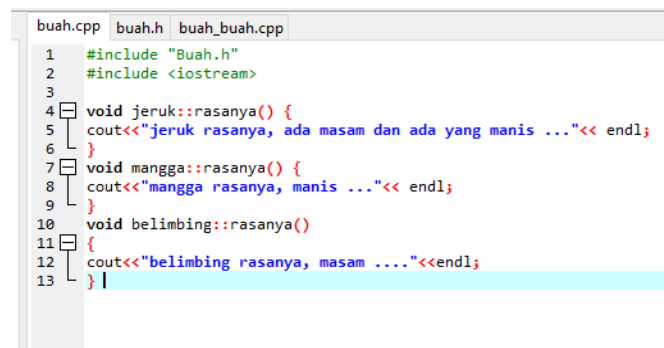
LAMPIRAN

INPUT :



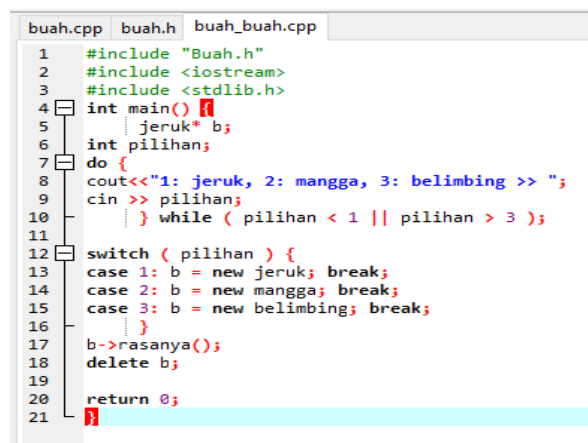
```
1 #ifndef BUAH_H
2 #define BUAH_H
3 using namespace std;
4 class jeruk
5 {
6 public:
7     virtual void rasanya();
8 };
9 class mangga : public jeruk{
10 public:
11     void rasanya();
12 };
13 class belimbing : public jeruk {
14 public:
15     void rasanya();
16 };
17 #endif
```

Gambar 2.1.1



```
1 #include "Buah.h"
2 #include <iostream>
3
4 void jeruk::rasanya() {
5     cout<<"jeruk rasanya, ada masam dan ada yang manis ..."<< endl;
6 }
7 void mangga::rasanya() {
8     cout<<"mangga rasanya, manis ..."<< endl;
9 }
10 void belimbing::rasanya()
11 {
12     cout<<"belimbing rasanya, masam ...."<<endl;
13 }
```

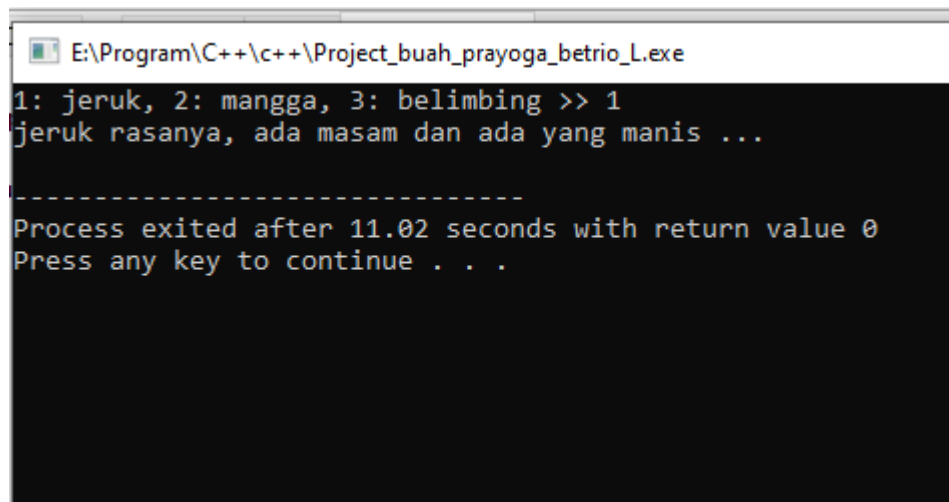
Gambar 2.1.2



```
1 #include "Buah.h"
2 #include <iostream>
3 #include <stdlib.h>
4 int main() {
5     jeruk* b;
6     int pilihan;
7     do {
8         cout<<"1: jeruk, 2: mangga, 3: belimbing >> ";
9         cin >> pilihan;
10    } while ( pilihan < 1 || pilihan > 3 );
11
12    switch ( pilihan ) {
13        case 1: b = new jeruk; break;
14        case 2: b = new mangga; break;
15        case 3: b = new belimbing; break;
16    }
17    b->rasanya();
18    delete b;
19
20    return 0;
21 }
```

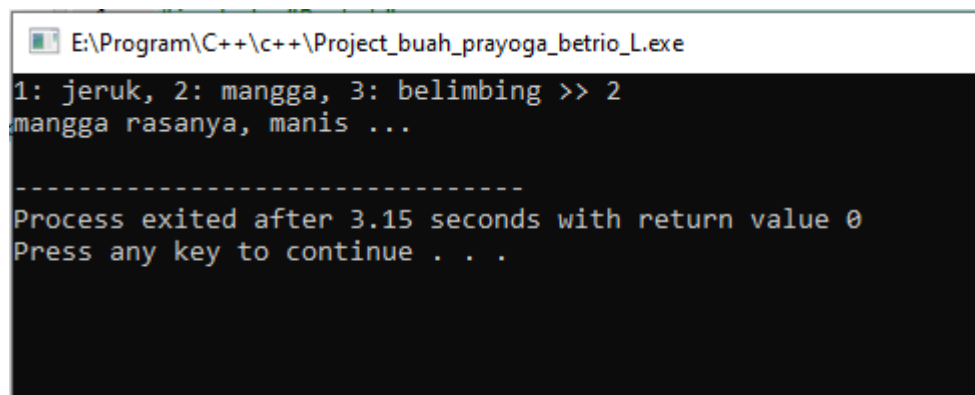
Gambar 2.1.3

OUTPUT :



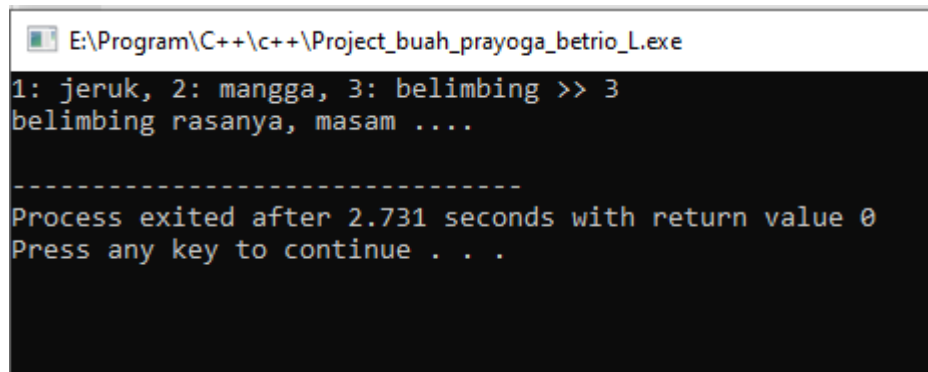
```
E:\Program\C++\c++\Project_buah_prayoga_betrio_L.exe
1: jeruk, 2: mangga, 3: belimbing >> 1
jeruk rasanya, ada masam dan ada yang manis ...
-----
Process exited after 11.02 seconds with return value 0
Press any key to continue . . .
```

Gambar 2.1.4



```
E:\Program\C++\c++\Project_buah_prayoga_betrio_L.exe
1: jeruk, 2: mangga, 3: belimbing >> 2
mangga rasanya, manis ...
-----
Process exited after 3.15 seconds with return value 0
Press any key to continue . . .
```

Gambar 2.1.5



```
E:\Program\C++\c++\Project_buah_prayoga_betrio_L.exe
1: jeruk, 2: mangga, 3: belimbing >> 3
belimbing rasanya, masam ....

-----
Process exited after 2.731 seconds with return value 0
Press any key to continue . . .
```

Gambar 2.1.6