LAPORAN HASIL PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN II

NAMA : **REMEMBER SITOMPUL**
NIM : **193020503022**
KELAS : **A**
MODUL : **PEMROGRAMAN BERORIENTASI OBJEK**

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PALANGKA RAYA
2020

LAPORAN HASIL PRAKTIKUM

ALGORITMA DAN PEMROGRAMAN II

Nama : REMEMBER SITOMPUL

NIM : 193020503022

Kelas : A

Modul : PEMROGRAMAN BERORIENTASI OBJEK

Komposisi	MAX	Nilai
BAB I Tujuan dan Landasan Teori	10	7
BAB II Pembahasan	60	50
BAB III Kesimpulan	20	13
Daftar Pustaka	5	3
Lampiran	5	3
Jumlah	100	76

Penilai

Asisten Praktikum

Diana

BAB I

TUJUAN DAN LANDASAN TEORI

1. TUJUAN
   - Memahami dasar dasar pemrograman berorientasi objeck
   - Memahami enkapsulasi
   - Membuat kelas dan objeck

2. DASAR TEORI

Pemrograman Object Oriented(OO) adalah suatu konsep pemrograman yang menggunakan objek sebagai building block dalam pengembangan aplikasinya(Weisfeld, 2008). Dalam pemrograman OO, objek memiliki data dan behaviour. Data adalah atribut yang melekat pada objek, yang akan merepresentasikan state objek tersebut. Sedangkan behaviour adalah fungsi-fungsi yang dapat dilakukan oleh objek, yang selanjutnya merepresentasikan sifat-sifat atau perilaku objek. Objek didefinisikan dalam suatu Kelas. Kelas inilah yang bertindak sebagai cetak-biru dalam pembentukan suatu objek. Objek dari suatu kelas akan mengikuti kontrak yang tertulis dalam definisi kelas tersebut, kontrak itu meliputi data dan behaviour yang didefinisikan kelas sehingga objek dapat disebut juga sebagai instance of a class. Pemrograman OO memiliki beberapa konsep dasar yang digunakan dalam pengembangan aplikasi berbasis OO. Konsepkonsep tersebut antara lain adalah encapsulation, inheritance, polymorphism, dan composition.

1.1 Encapsulation

Sebuah objek memiliki atribut dan fungsi yang merepresentasikan state dan behaviour-nya. Encapsulation 6 7 adalah konsep menyembunyikan atribut suatu objek dari objek lain. Komunikasi antar objek harus dilakukan melalui antarmuka yang telah ditentukan dalam definisi/kontrak yang tertulis dalam Kelas. Antarmuka ini berupa fungsifungsi yang didefinisikan oleh kelas. Sebagai contoh, Objek dari kelas A memiliki atribut x, atribut ini hanya dapat diakses oleh objek lain melalui fungsi setX() dan getX() yang didefinisikan oleh kelas A. Dengan begitu objek lain tidak perlu mengetahui detail implementasi yang diterapkan kelas A untuk mengakses data atribut x yang dimilikinya. Objek lain cukup

mengetahui fungsi apa yang disediakan kelas A untuk mengakses atribut tersebut.

1.2 Inheritance

Keuntungan dari pemrograman OO adalah kemampuan code reuse, memanfaatkan kode yang telah ditulis untuk kegunaan lain, inheritance adalah salah satu penerapannya. Inheritance adalah suatu cara untuk membentuk Kelas baru, berdasarkan Kelas yang telah dibentuk. Kelas baru ini nantinya akan mewarisi atribut dan fungsi yang terdefinisi dalam Kelas yang telah dibentuk sebelumnya. Kelas baru yang terbentuk ini selanjutnya disebut sebagai subclass atau kelas anak. Sedangkan Kelas pembentuk subclass disebut sebagai superclass atau kelas induk. Atribut dan fungsi-fungsi yang diwarisi kelas anak dari kelas induknya dapat juga diubah atau ditambah. Ini menjadikan kelas anak sebagai versi kelas induk yang memiliki perilaku dan atribut yang lebih khusus. Selain mewarisi atribut dan fungsi kelas induk, kelas anak juga mewarisi type dari kelas induk. Sehingga kelas 8 anak memiliki is-a relationship dengan kelas induk. Contohnya, Kelas Segitiga dan Lingkaran merupakan kelas yang dibentuk dari Kelas Bentuk2D. Kelas Segitiga dan Lingkaran akan mewarisi atribut dan fungsi dari Kelas Bentuk2D. Selain itu, Segitiga dan Lingkaran dapat dikatakan sebagai Bentuk2D, dengan kata lain ’is-a’ Bentuk2D. Objek Segitiga memiliki type kelas Segitiga dan juga type kelas induknya, yaitu Bentuk2D. Ini yang selanjutnya disebut sebagai polymorphism(bahasa yunani: memiliki banyak bentuk).

1.3 Polymorphism

Polymorphism berkaitan erat dengan konsep inheritance, suatu kelas dapat memiliki banyak bentuk(polimorfis) dengan mewarisi bentuk atau type dari kelas induknya melalui inheritance.

Kelas

Kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebagai contoh 'class of dog' adalah suatu unit yang terdiri atas definisi-definisi data dan fungsi-fungsi yang menunjuk pada berbagai macam perilaku/turunan dari anjing. Sebuah class adalah dasar dari modularitas dan struktur dalam pemrograman berorientasi object. Sebuah class secara tipikal sebaiknya dapat dikenali oleh seorang non-programmer sekalipun terkait dengan domain permasalahan yang ada, dan kode yang terdapat dalam sebuah class sebaiknya (relatif) bersifat mandiri dan

independen (sebagaimana kode tersebut digunakan jika tidak menggunakan OOP). Dengan modularitas, struktur dari sebuah program akan terkait dengan aspek-aspek dalam masalah yang akan diselesaikan melalui program tersebut. Cara seperti ini akan menyederhanakan pemetaan dari masalah ke sebuah program ataupun sebaliknya.

Objek

Membungkus data dan fungsi bersama menjadi suatu unit dalam sebuah program komputer; objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek.

Dalam konteks “Pemrograman Berorientasi obyek” Obyek adalah orang, tempat, benda, kejadian atau konsep - konsep yang ada di dunia nyata yang penting bagi suatu aplikasi (perangkat lunak dan / atau sistem informasi).

Pemahaman obyek memiliki dua fungsi, yaitu :

1. Memudahkan untuk mempelajari secara seksama hal - hal di dunia nyata.
2. Menyediakan suatu dasar yang kuat dalam implementasi ke dalam sistem terkomputerisasi.

Abstraksi

Kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari "pelaku" abstrak yang dapat melakukan kerja, laporan dan perubahan keadaannya, dan berkomunikasi dengan objek lainnya dalam sistem, tanpa mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak, dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan.

Enkapsulasi

Memastikan pengguna sebuah objek tidak dapat mengganti keadaan dalam dari sebuah objek dengan cara yang tidak layak; hanya metode dalam objek tersebut yang diberi izin untuk mengakses keadaannya. Setiap objek mengakses interface yang menyebutkan bagaimana objek lainnya dapat berinteraksi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut.

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 pesa tersebut dikirim.

Contohnya: bila sebuah burung menerima pesan "gerak cepat", dia akan menggerakan 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 tungal 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.

Bahasa pemrograman yang mendukung OOP antara lain:

1. Visual Foxpro
2. Java
3. C++
4. Pascal (bahasa pemrograman)
5. SIMULA
6. Smalltalk
7. Ruby
8. Python
9. PHP
10. C#
11. Delphi
12. Eiffel
13. Perl
14. Adobe Flash AS 3.0

BAB II
PEMBAHASAN

1. Program pertama

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

class burung
{
  public:
      burung(int);
      int jlhSayap();
      int jlhKaki();


  private:
      int Sayap;
      int Kaki;

};

  burung::burung(int s){
    Sayap=s;
    Kaki=s;

 }

 int burung::jlhSayap(){
   return Sayap;
 }

 int burung::jlhKaki(){
   return Kaki;
 }

 int main(){
   burung s(2);
   burung k(2);


   cout<<"        BURUNG ELANG       "<<endl;
   cout<<"==================================="<<endl;
   cout<<"jumlah sayap burung adalah : "<<s.jlhSayap()<<endl;
   cout<<"jumlah kaki burung adalah  : "<<k.jlhKaki()<<endl;
   cout<<"jenis burung pemakan       : daging "<<endl;
  return 0;
 }
```

Program pertama menginputkan kelas BURUNG ELANG

```
#include <iostream>.
```

kode #include <iostream>.

Tanda '#' disebut *preprocessor directive*.

Preprocessor directive adalah perintah – perintah yang diberikan kepada compiler untuk melakukan definisi, misalnya untuk memasukkan file library, dan lain sebagainya.

Jika kita lihat program diatas kita akan memasukkan (include) library iostream ke dalam program. iostream adalah header yang dibutuhkan untuk "kegiatan" input dan output.

```
using namespace std;
```

Fungsinya supaya kita tidak perlu mengetikan std::cout untuk mencetak output namun hanya cukup menggunakan fungsi cout saja, karena telah menggunakan using namespace std;

```
class burung
{
  public:
       burung(int);
       int jlhSayap();
       int jlhKaki();



  private:
       int Sayap;
       int Kaki;


};
  burung::burung(int s){
    Sayap=s;
    Kaki=s;
```

```
}


int burung::jlhSayap(){
  return Sayap;
}


int burung::jlhKaki(){
  return Kaki;
}
```

Public adalah fungsi yang dapat diakses oleh umum atau dapat diakses oleh siapa saja.

Privat merupakan fungsi yang hanya dapat diakses secara internal oleh objek.

Dalam badan ***Class*** program tersebut, ***Private*** yang dapat diaskes oleh program secara khusus adalah *int sayapl; int kaki;* jadi maksudnya bahwa lokal ini hanya bleh diskses secara khusus saja. Kemudian dalam badan program terdapat ***publicc*** 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 mengisikan data yang dimasukan dalam program ke tempatnya, seperti *Int jumlah sayap;* yang mengartikan bahwa isi dari *sayap* akan disisi data dari sayap. Dan bentuk *int jlh kaki;* berguna untuk menjadikan parameter dari yang diambil dari *private*.

Kemudian dalam // untuk menampilakan digunakan untuk menjadi konstruktor atau pembentuk dalam outputnya. Disana terdapat int peroleh Info yang menjadi konstruktornya. Dalam badan Class tersebut akan menampilakan outputnya.

```
int main(){
   burung s(2);
   burung k(2);
```

```
  cout<<"        BURUNG ELANG       "<<endl;
  cout<<"===================================="<<endl;
  cout<<"jumlah sayap burung adalah : "<<s.jlhSayap()<<endl;
  cout<<"jumlah kaki burung adalah  : "<<k.jlhKaki()<<endl;
  cout<<"jenis burung pemakan       : daging "<<endl;
 return 0;
}
```

Int main() adalah fungsi utama dari sebuah kode bahasa C++. Fungsi ini memberikan nilai balik menurut type datanya, dan karena memiliki nilai balik maka diberikan perintah return nilai. Int main() artinnya main program mengembalikan nilai int secara default, int main() akan mengembalikan nilai 0, dan fungsi main() tidak memiliki bagan deklarasi lokal, dan hanya memiliki sebuah pernyataan yang dapat dieksekusi, berupa fungsi output printf().

Setelah membuat int main sebagai pemberi nilai pada badan program ini terdapat cout yang berfungsi menampilkan output pada program burung elang di atas. Contohnya, cout<<"jumlah sayap burung adalah : "<<s.jlhSayap()<<endl;,, yang nantinya akan menampilakan output program yang bertuliskan “jumlah sayap burung adalah : (sesuai yang sudah di cantumkan dalam program tersebut).

2. Program dua ini sama seperti program pertama membuat class hewan, di program kedua ini menggunakan hewan yang berbeda yaitu hewan katak.

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

class katak
{
  public:
      katak(int);
      int jlhKaki();
      int jlhMata();


  private:
      int kaki;
      int mata;
```

```
};

 katak::katak(int k){
   kaki=k;
   mata=k;
}


int katak::jlhKaki(){
  return kaki;
}

int katak::jlhMata(){
  return mata;
}

int main(){
  katak K(4);
  katak M(2);


  cout<<"        HEWAN KATAK        "<<endl;
  cout<<"===================================="<<endl;
  cout<<"jumlah kaki normal     : "<<K.jlhKaki()<<endl;
  cout<<"jumlah mata normal     : "<<M.jlhMata()<<endl;
  cout<<"makanan                : serangga"<<endl;
  cout<<"------------------------------------"<<endl;
 return 0;
}
```

Program kedua menginputkan class KATAK

```
#include <iostream>.
```

kode #include <iostream>.

Tanda '#' disebut *preprocessor directive*.

Preprocessor directive adalah perintah – perintah yang diberikan kepada compiler untuk melakukan definisi, misalnya untuk memasukkan file library, dan lain sebagainya.

Jika kita lihat program diatas kita akan memasukkan (include) library iostream ke dalam program. iostream adalah header yang dibutuhkan untuk "kegiatan" input dan output.

```
using namespace std;
```

Fungsinya supaya kita tidak perlu mengetikan std::cout untuk mencetak output namun hanya cukup menggunakan fungsi cout saja, karena telah menggunakan using namespace std;

```
class katak
{
  public:
      katak(int);
      int jlhKaki();
      int jlhMata();


  private:
      int kaki;
      int mata;

};

  katak::katak(int k){
    kaki=k;
    mata=k;
 }


 int katak::jlhKaki(){
   return kaki;
 }

 int katak::jlhMata(){
   return mata;
 }
```

Public adalah fungsi yang dapat diakses oleh umum atau dapat diakses oleh siapa saja.

Privat merupakan fungsi yang hanya dapat diakses secara internal oleh objek.

Dalam badan ***Class*** program tersebut, ***Private*** yang dapat diaskes oleh program secara khusus adalah *int kaki; int matai;* jadi maksudnya bahwa lokal ini hanya bleh diskses secara khusus saja. Kemudian dalam badan program terdapat ***publicc*** 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 mengisikan data yang dimasukan dalam program ke tempatnya, seperti *Int jumlah kaki;* yang

mengartikan bahwa isi dari *kaki* akan disisi data dari sayap. Dan bentuk *int jlh mata;* berguna untuk menjadikan parameter dari yang diambil dari *private*.

Kemudian dalam // untuk menampilakan digunakan untuk menjadi konstruktor atau pembentuk dalam outputnya. Disana terdapat int peroleh Info yang menjadi konstruktornya. Dalam badan Class tersebut akan menampilakan outputnya.

```
int main(){
   katak K(4);
   katak M(2);


   cout<<"         HEWAN KATAK         "<<endl;
   cout<<"=====================================">>endl;
   cout<<"jumlah kaki normal     : "<<K.jlhKaki()<<endl;
   cout<<"jumlah mata normal     : "<<M.jlhMata()<<endl;
   cout<<"makanan                : serangga"<<endl;
   cout<<"-------------------------------------"<<endl;
  return 0;
 }
```

Int main() adalah fungsi utama dari sebuah kode bahasa C++. Fungsi ini memberikan nilai balik menurut type datanya, dan karena memiliki nilai balik maka diberikan perintah return nilai. Int main() artinnya main program mengembalikan nilai int secara default, int main() akan mengembalikan nilai 0, dan fungsi main() tidak memiliki bagan deklarasi lokal, dan hanya memiliki sebuah pernyataan yang dapat dieksekusi, berupa fungsi output printf().

Setelah membuat int main sebagai pemberi nilai pada badan program ini terdapat cout yang berfungsi menampilkan output pada program hewan katak di atas. Contohnya, cout<<"jumlah kaki normal:"<<K.jlhKaki()<<endl;,, yang nantinya akan menampilakan output program yang bertuliskan “jumlah sayap burung adalah : (sesuai yang sudah di cantumkan dalam program tersebut).

gambar hasil

Output

BAB III

KESIMPULAN

Class adalah salah satu dari konsep OOP yang digunakan untuk membungkus data abstraksi *procedural* sebagai deskripsi tergeneralisir atau rancangan dari sebuah *object* untuk mendefinisikan atau menggambarkan isi dan tingkah laku sebagai entitas dari *object.*

Objek adalah membungkus data dan fungsi bersama menjadi suatu unit dalam sebuah program komputer; objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek.

Class dan *Object* merupakan fitur yang sangat membantu untuk mendirikan sebuah program besar, menjadikan sebuah code program yang ditulis oleh programmer mudah untuk dimengerti, lebih terstruktur, dan juga mudah dalam pemeliharaan program.

Acces modifier memiliki 3 tipe yaitu *public, private,* dan *protected.* Semua system terdiri dari class-class dan objek. Suatu system dicapai melalui kerjasama antar object, interaksi antar object disebut object relationship. Method merepresentasikan operasi-operasi yang dapat dilakukan oleh objek. Yang membedakan antara variabel dan method adalah method selalu diakhiri dengan ( ) atau {}.

Constructor merupakan suatu method yang akan memberikan nilai awal dari pada saat suatu objek dibuat.

DAFTAR PUSTAKA

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

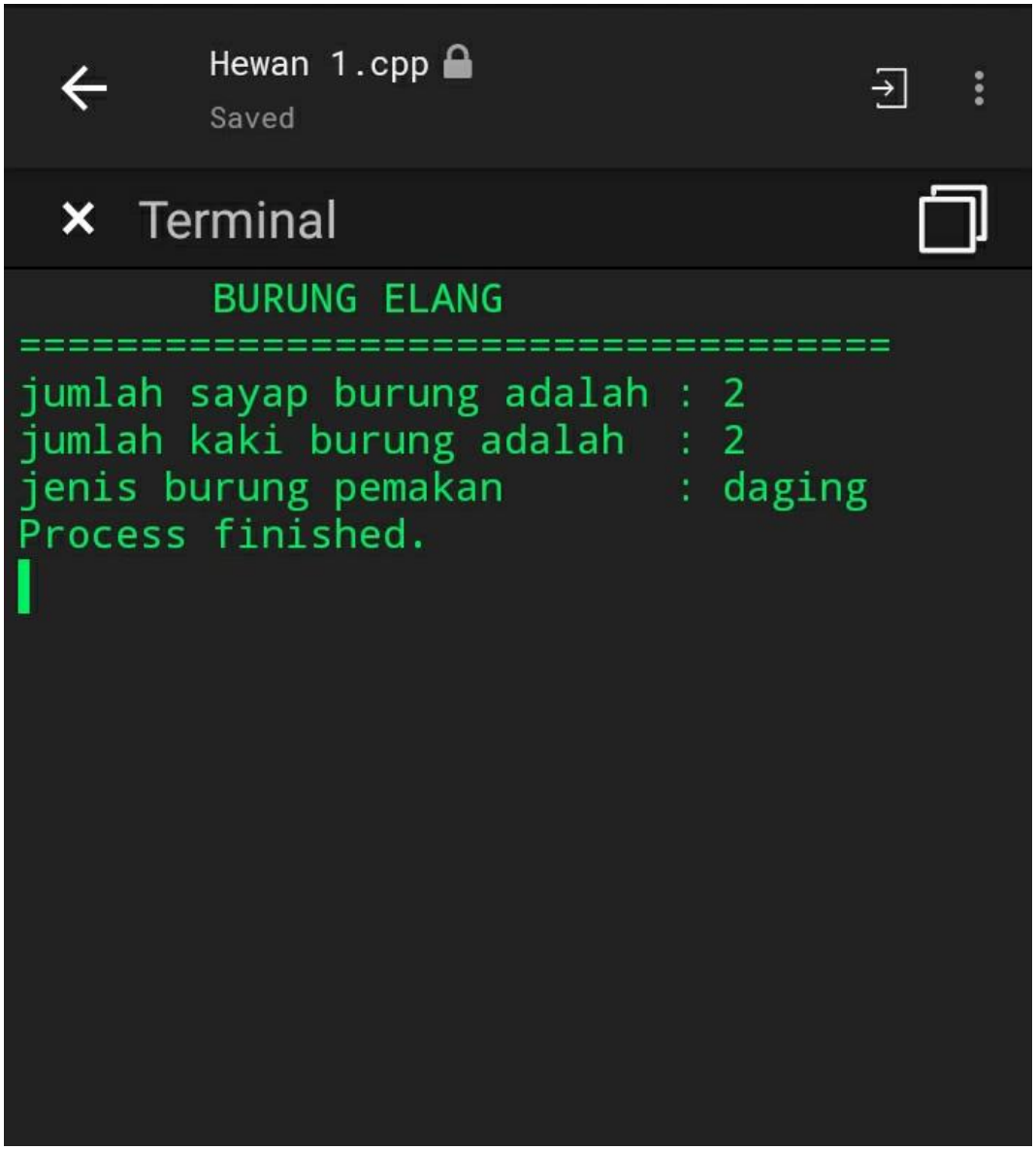
http://rendyandrians.blogspot.com/2014/08/perbedaan-int-main-dan-void-main.html. Diakses 8 April 2020

https://www.sinauarduino.com/artikel/pengertian-classes-pada-pemrograman-cpp/. Diakses 8 April 2020

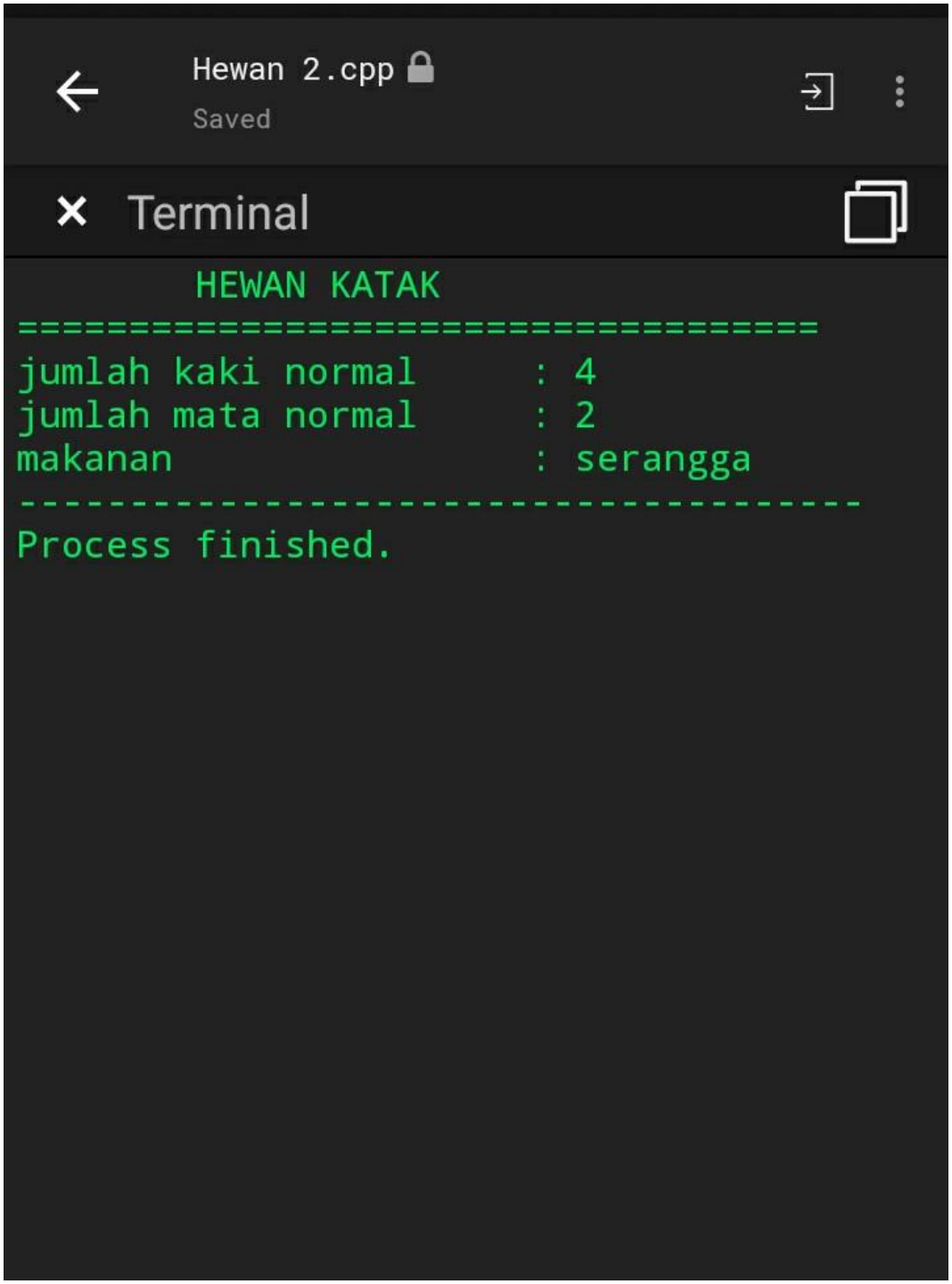
dicek lagi aturan penulisannya

LAMPIRAN

Tampilan output program pada C++



Nomor gambar


Hewan 2.cpp
Saved
Terminal
HEWAN KATAK
==========
jumlah kaki normal : 4
jumlah mata normal : 2
makanan : serangga

Process finished.