1. **Define basic algorithms to carry out an operation and outline the process of programming an application.**
   1. **Define an algorithm and outline the process in building an application**
2. Thuật toán là gì? Tầm quan trọng của thuật toán?

* Thuật toán (Algorithm) hiểu đơn giản là một phương pháp để giải quyết một bài toán khó. Và có thể hiểu là một tập hợp hữu hạn bao gồm các hướng dẫn để xác định rõ ràng, và có thể thực hiện được bằng máy tính.
* Tầm quan trọng của thuật toán:
  + Thuật toán có vai trò rất quan trọng trong lĩnh vực máy tính ngày nay, và cũng là một nền tảng cho mọi ứng dụng và lĩnh vực về công nghệ thông tin(Information Technology)

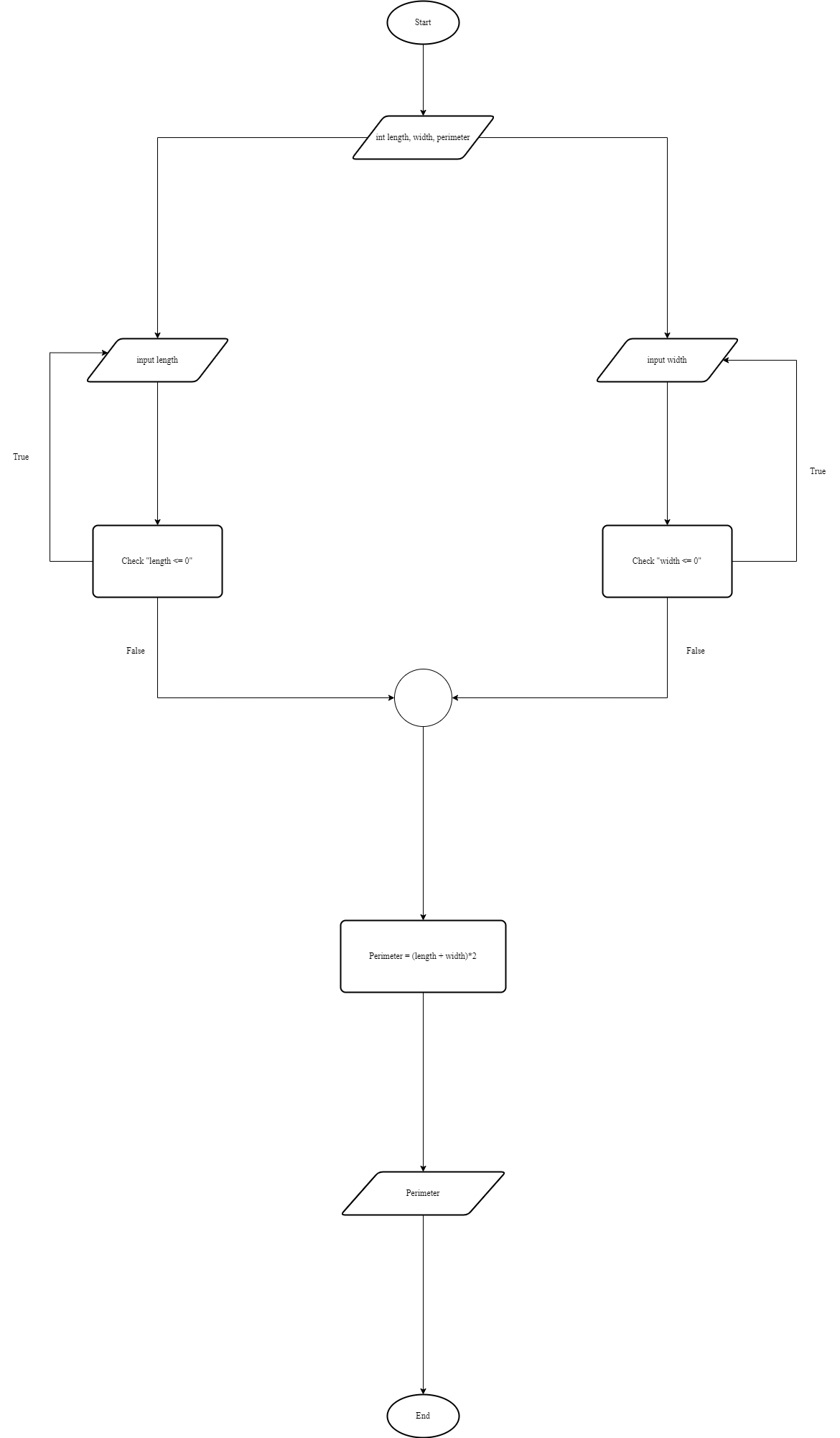
1. Trình bày các cách biểu diễn thuật toán?

* C1: Sử dụng ngôn ngữ tự nhiên
* C2: Sử dụng lưu đồ flowchart
* C3: Sử dụng mã giả

1. Trình bày các bước xây dựng và phát triển phần mềm (Vòng đời phát triển sản phẩm):

* Khảo sát và phân tích (Analysis):
  + Giai đoạn này là bước đầu tiên trong quá trình phát triển phần mềm. Tại đây, chúng ta thu nhập và xác định rõ các yêu cầu của người dùng và các bên liên quan đối với sản phẩm phần mềm sắp xây dựng.
* Thiết kế (Design):
  + Từ các yêu cầu và thông số kỹ thuật đã xác định, chúng ta vạch ra kiến trúc tổng hợp cần thiết để tạo ra phần mềm.
  + Các yếu tố như ngân sách, thời gian, công nghệ áp dụng, và mức độ rủi ro cũng được xác định rõ rang.
* Phát triển (Development):
  + Giai đoạn này là quá trình xây dựng phần mềm dựa trên các đặc tả đã ghi ra.
  + Lập trình viên viết mã nguồn, kiểm tra và tích hợp các thành phần.
* Kiếm thử (Testing)
  + Giai đoạn này, chúng ta kiểm tra phần mềm để đảm bảo tính đúng đắn, hiệu suất và bảo mật.
  + Kiểm thử có thể bao gồm kiểm thử đơn vị, kiểm thử tích hợp, kiểm thử hệ thống và kiểm thử chấp nhận.
* Triển khai và bảo trì (Deployment and Maintenance):
  + Sau khi phần mềm đã qua kiểm thử, chúng ta triển khai nó vào môi trường thực tế
  + Bảo trì phần mềm đảm bảo rằng nó luôn hoạt động ổn định và được cập nhận khi cần thiết.
  1. **Determine the steps taken from writing code to execution.**
* Từ các bước phân tích ở phần 1.1, sinh viên áp dụng để viết chương trình phần mềm tính chu vi hình chữ nhật, với chiều dài và chiều rộng là số lớn hơn 0 được nhập từ bàn phím.

Flow chart



Source code

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int length\_of\_rectangle, width\_of\_rectangle, perimeter\_rectangle;

do {

System.out.print("The width of the rectangle: ");

width\_of\_rectangle = input.nextInt();

if (width\_of\_rectangle<= 0) {

System.out.println("The width of the rectangle should be greater than 0.");

}

} while (width\_of\_rectangle <= 0);

do {

System.out.print("The length of the rectangle: ");

length\_of\_rectangle = input.nextInt();

if (length\_of\_rectangle <= 0) {

System.out.println("The length of the rectangle should be greater than 0.");

}

} while (length\_of\_rectangle <= 0);

perimeter\_rectangle = (length\_of\_rectangle + width\_of\_rectangle) \* 2;

System.out.println("The perimeter of the rectangle is: " + perimeter\_rectangle);

}

}

Kết quả

A screenshot of a computer

Description automatically generated

**1.3: Analyse the process of writing code, including the potential challenges**

* Bạn thường gặp khó khăn gì ở những bước nào trong quá trình viết code, nêu chi tiết từng vấn đề?
  + Debug (Sửa lỗi)
    - Khi viết code xong và bắt đầu kiểm tra thì sẽ gặp trường hợp code báo lỗi, nhưng không biết mình sai ở vị trí nào.

* 1. **Evaluate the implementation of an algorithm in a suitable language and the relationship between the written algorithm and the code variant.**
* Tại sao thuật toán và ngôn ngữ lập trình độc lập với nhau?
  + Bởi vì chúng có mục đích hoạt động khác nhau:
    - Thuật toán tập trung vào cách giải quyết vấn đề và các thực hiện nhiệm vụ.
    - Ngôn ngữ lập trình tập trung vào việc biểu diễn ý tưởng và mã nguồn để máy tính hiểu.
* Chọn 1 trong các bài dưới đây, áp dụng giải thuật và viết code trên 2 ngôn ngữ lập Java, PHP.
  + Nhập 3 số tự nhiên a, b, c từ bàn phím. Tìm số lớn nhất và in kết quả ra màn hình.

Java

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int a, b, c, max;

System.out.println("Enter the first number: ");

a = scanner.nextInt();

System.out.println("Enter the second number: ");

b = scanner.nextInt();

System.out.println("Enter the third number: ");

c = scanner.nextInt();

System.out.println();

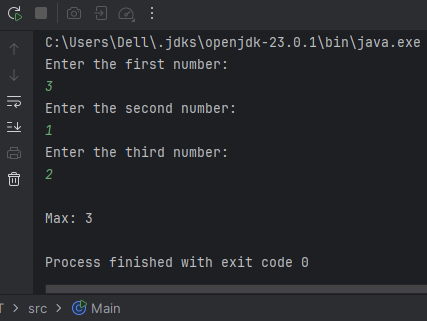
max = Math.max(a, b);

max = Math.max(max, c);

System.out.println("Max: " + max);

}

}



PHP

<?php

do{

$a = readline("Enter the first number: ");

$a = (int) $a;

$b = readline("Enter the second number: ");

$b = (int) $b;

$c = readline("Enter the third number: ");

$c = (int) $c;

$max = $a;

if($max < $b){

$max = $b;

}

if ($max < $c){

$max = $c;

}

echo "Numbers entered successfully!";

echo "Max: " . $max . PHP\_EOL;

$continue = readline("Do you want to enter another set of number? (yes/no): ");

} while (strtolower($continue) == 'yes');

echo "End" . PHP\_EOL;

?>

A screen shot of a computer

Description automatically generated

1. Explan the characteristics of procedural, object orientated and event-driven programming
   1. **Discuss what procedural, objectorientated and eventdriven paradigms are; their characteristics and the relationship between them.**

* Trình bày khái niệm và đặc điểm của 3 hướng lập trình: Thủ tục, đối tượng, sự kiện?
  + Lập trình thủ tục (Procedural Programming)
    - Khái niệm:
      * Lập trình hướng thủ tục là lập trình tập trung vào việc chia nhỏ một chương trình lớn thành các khối nhỏ hơn, đặc biệt hướng lập trình này là mô hình đầu tiên và phổ biến.
    - Đặc điểm:
      * Tập trung vào các thủ tục
      * Sử dụng dòng điều khiển tuyên tính, trừ một số câu lệch như if, else, for, while.
      * Sử dụng dữ liệu toàn cục
      * Luôn tương tác với người dùng
  + Lập trình hướng đối tượng (Object-Oriented Programming – OOP)
    - Khái niệm:
      * Là một mô hình lập trình tổ chức thiết kế phần mềm dữ liệu xung quanh. OOP coi mọi thứ trong chương trình đều là một đối tượng thay vì tập trung vào các hàm và những thứ logic như các lập trình truyền th
    - Đặc điểm:
      * Tính đóng gói (Encapsulation): ẩn đi các chi tiết thực hiện bên trong một đối tượng, bảo vệ dữ liệu khỏi bị truy cập và bị sửa đổi một cách không kiểm soát.
      * Tính kế thừa (Inheritance): Tạo ra các lớp con kế thừa phương thức và thuộc tính của các lớp cha, tạo ra mối quan hệ “là một” giữa các lớp.
      * Tính đa hình (Polymorphism): Một đối tượng có thể chia ra nhiều hình dạng khác nhau, cho phép sử dụng các đối tượng khác trong cùng một giao diện
      * Tính trừu tượng (Abstraction): Tập trung vào các hành vi quan trọng, bỏ qua các chi tiết không cần thiết của đối tượng
  + Lập trình hướng sự kiện (Event-Driven Programming)
    - Khái niệm:
      * Là một mô hình lập trình trong đó dòng chảy của chương trình được điều khiển bởi các sự kiện xảy ra, thay vì cố định như lập trình thủ tục
    - Đặc điểm:
      * Chương trình phản ứng với sự kiện đang xảy ra như việc chờ hoặc xử lý các sự kiện
      * Sự kiện là trung tâm
      * Vòng lặp sự kiện
      * Có tính linh hoạt cao
      * Phù hợp với giao diện người dùng
* So sánh ưu điểm, nhược điểm của lập trình hướng thủ tục, hướng sự kiện
  + Lập trình hướng thủ tục (Procedural Programming):
    - Ưu điểm:
      * Dễ hiểu và dễ quản lý các chương trình
      * Có hiệu xuất cao cho các tác vụ
    - Nhược điểm:
      * Khó mở rộng và bảo trì khi chương trình trở nên phức tạp
      * Không linh hoạt trong việc xử lý
  + Lập trình hướng sự kiện (Event – Driven Programming)
    - Ưu điểm:
      * Có tính linh hoạt và dễ bảo trì, mở rộng
      * Phù hợp với giao diện người dùng, hệ thống nhúng và các ứng dụng cần tương tác
      * Tính phản hồi cao
    - Nhược điểm:
      * Phức tạp hơn với các chương trình nhỏ
      * Quản lý lường có thể khó khăn hơn
* Trong một chương trình, phần mềm thường kết hợp nhiều hướng lập trình? Giải thích vì sao?
  + Trong thực tế, một chương trình và phần mềm thường kết hợp với nhiều hướng lập trình như vì:
    - Để tối ưu hóa chương trình
    - Tăng tính linh hoạt
    - Tận dụng các thư viện và công cụ có sẵn
  1. **Compare the procrdural, object-orientated and eventdriven paradigms used in given source code of an application.**
* Viết 1 chương trình “Cuộc phiêu lưu của Mango: Hãy tưởng tượng có một chú khỉ con tinh nghịch tên là Mango đang đu đưa từ cảnh này sang cảnh khác để thu thập những quả chuối xuất hiện trên đường đi. Người điều khiển chương trình bấm phím Space để thu thập những quả chuối. Hãy viết một đoạn chương trình bằng java để mô tả cuộc phiêu lưu của Mango”.
  + Source code:

Mango.java

public class Mango {  
 private String name;  
 private int bananasCollected;  
  
  
 public Mango(String name) {  
 this.name = name;  
 this.bananasCollected = 0;  
 }  
  
  
 public void collectBanana() {  
 bananasCollected++;  
 System.*out*.println(name + " đã thu thập được 1 quả chuối! Tổng chuối: " + bananasCollected);  
 }  
  
  
 public int getBananasCollected() {  
 return bananasCollected;  
 }  
}

Game.java

import java.util.Scanner;  
  
public class Game {  
 private Mango mango;  
 private Scanner scanner;  
  
  
 public Game() {  
 mango = new Mango("Mango");  
 scanner = new Scanner(System.*in*);  
 }  
  
  
 public void start() {  
 String input = "";  
  
  
 System.*out*.println("Cuộc phiêu lưu của Mango bắt đầu!");  
 System.*out*.println("Gõ 'space' để thu thập chuối (gõ 'exit' để kết thúc).");  
  
  
 while (!input.equalsIgnoreCase("exit")) {  
 System.*out*.print("Nhập lệnh: ");  
 input = scanner.nextLine();  
  
  
 if (input.equalsIgnoreCase("space")) {  
 mango.collectBanana();  
 } else if (!input.equalsIgnoreCase("exit")) {  
 System.*out*.println("Hãy gõ 'space' để thu thập chuối hoặc 'exit' để thoát.");  
 }  
 }  
  
  
 endGame();  
 }  
  
   
 private void endGame() {  
 System.*out*.println("Cuộc phiêu lưu kết thúc. Tổng số chuối Mango đã thu thập: " + mango.getBananasCollected());  
 scanner.close();  
 }  
}

MangoAdventure.java

public class MangoAdventure {  
 public static void main(String[] args) {  
  
 Game game = new Game();  
 game.start();  
 }  
}

Main.java

public class Main {  
 public static void main(String[] args) {  
 MangoAdventure.*main*(args);  
 }  
}

Kết quả:

A screenshot of a computer

Description automatically generated

* Sinh viên cần giải thích: Phần nào là thủ tục, đối tượng, phần nào là sự kiện và tại sao?
  + Đối tượng
    - Mango là một đối tượng
    - Thuộc tính
      * name: Tên của chú khỉ
      * bananasCollected: Số lượng chuối đã thu thập
    - Hành vi:
      * collectBanana(): Phương thức để thu thập chuối, tăng biến bananasCollected lên 1.
      * getBananaCollected(): Phương thức để lấy tổng số chuối đã thu thập.
    - Giải thích: Vì nó có thuộc tính và hành vi riêng, mô phỏng một thực thể trong cuộc sống.
  + Thủ tục:
    - Các phương thức trong class Game được xem là thủ tục:
      * Start(): Thủ tục để bắt đầu trò chơi, bao gồm hướng dẫn và quản lý vòng lặp thu thập chuối.
      * endGame(): Thủ tục để kết thúc trò chơi và hiển thị tổng số chuối
    - Giải thích: Các phương thức này thực hiện các tác vụ cụ thể (bắt đầu trò chơi, kết thúc trò chơi) và có thể được gọi từ các phần khác trong chương trình.
  + Sự kiện
    - Các sự kiện là các lệnh mà người chơi nhập vào:
      * Gõ space để thu thập chuối.
    - Giải thích: Các lệnh mà người chơi nhập vào đều là sự kiện mà chương trình cần phản hồi. Các sự kiện này ảnh hưởng đến trạng thái của chương trình (thu thập chuối hoặc kết thúc trò chơi).
      1. **Critically evaluate the source code of an application that implements the procedural, object-orientated and event-driven paradigms, in terms of the code structure and characteristics. (D2)**
* Dựa vào