Discussion-1: Introduction to Programming

Discussion Topic:

Select only one of the following questions and offer a good, substantive reply. Some repetition is inevitable, but please aim to reply to a question that has not already been answered.

Please include the question number you respond to, so we all can better understand your post, such as "#1", "#2", etc.:

1. What are the principal components involved in developing a program? Describe the benefits of breaking up a program into small modules and provide an example of an application that makes the most of one of these benefits.
2. Describe the tasks performed by a software programmer. What skills are necessary and why? What impact do these skills have on the programs created?
3. What does the term algorithm mean? What is a flowchart? What is pseudocode? Why are flowcharts or pseudocode needed?

Problem-solving through computer programs is the main topic of our class. In this discussion, we dive into it by first analyzing the Python language and then exploring several tools that help us analyze the problem and then design the needed solution. Flowcharts, pseudocode, and IPO charts are these tools, and they allow us to specify an algorithm that will resolve the problem. Let's discuss all of this -- pick a question and shoot!

My Post:

Hello class,

For this discussion, I have chosen question #1, which is divided into two parts. The first part asks what the principal components are involved in developing a program, while the second part asks the benefits of breaking up a program into small modules, along with an example of an application that benefits from this approach.

What are the principal components involved in developing a program?

The principal components involved in developing an application are planning and analysis, design, implementation, testing, and maintenance. The process is cyclical, meaning that the components are treated as repeatable steps during the lifetime of the applications. This was not always the case, in the past most applications were sold on Compact Disks (CDs). Software maintenance was only accessible to businesses that could afford to have a team of programmers capable of maintaining (upgrading/updating) their software. This application development model was not accessible to individual users or smaller businesses. Today, most individual user applications are developed using a programming development cycles model.

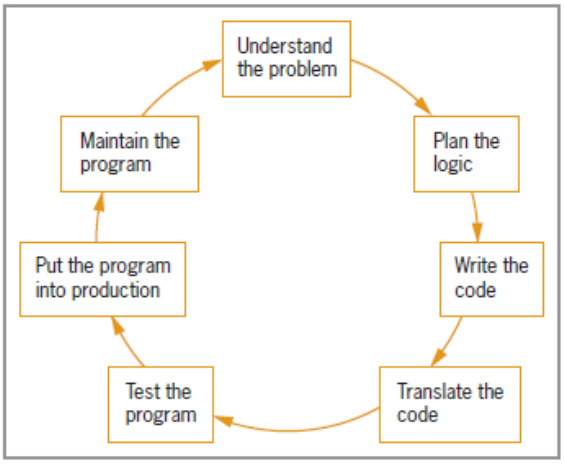
The following list outlines six components or steps involved in a programming development cycles model.

1. Analyze the problem or need: Understand the issue or need and decide on a programmatic solution.
2. Design the program (logic): Use tools like flowcharts to visualize the program's flow.
3. Code the program: Write the source code using a programming language.
4. Debug the program (test): Identify and fix errors or "bugs" in the code.
5. Formalize the solution: Ensure the program is ready for release and formalize documentation for understanding and future maintenance.
6. Release and maintain the program.

(Nguyen, 2019)

The word “cycles” is pluralized in the expression “programming development cycles” because the development steps can be repeated by section. For example, the steps ‘Code the program’ and ‘Debug the program’ can form a cyclic section of the development, meaning that after you debug the program you may need to recode the program and then debug it again.

The diagram below shows the cyclical nature of a programming development cycles model.
Figure 1



Note. From *Programming Development Cycles*, by Nguyen, 2019

Describe the benefits of breaking up a program into small modules and provide an example of an application that makes the most of one of these benefits.

Breaking up a program into small modules has several benefits such as readability, manageability, easier testing, reusability, and maintenance of code. Block-structured languages structurally implement what can be considered a low level of modulation for readability and functionality. For example, c and c++ use brackets {} to groups (modularized) and Python utilizes indentation for the same purpose (Klein, 2022). Object-oriented programming languages take it a step further by implementing classes that allow the

creation of object instances that encapsulate both code and data allowing modulation of a program. I store my program classes in different files and directories. Additionally, importing libraries, for example, in c++ and Python is a form of modular programming.

Modular programming is:

“Modular programming is a general programming concept where developers separate program functions into independent pieces. These pieces then act like building blocks, with each block containing all the necessary parts to execute one aspect of functionality.” (Macdonald, 2023)

“Modular programming is a software design technique that emphasizes separating the functionality of a program into independent, interchangeable modules, such that each contains everything necessary to execute only one aspect of the desired functionality.” (Busbee & Braunschweig, 2018)

The benefits of breaking up a program into small modules or modular programming can be listed as follows:

- Readability: Modular code is easier to read and understand because it's divided into logical sections, each performing a distinct function.
- Manageability: Smaller, self-contained modules are easier to manage because changes in one module are less likely to impact others.
- Easier Testing: Modules can be tested independently, making it simpler to isolate and resolve defects.
- Reusability: Functions or classes defined in one module can be reused in other parts of the program or in future projects, saving development time.
- Maintenance: Updating a module for new requirements or fixing bugs is more straightforward when the application is modularized, enhancing long-term code maintenance.

(Busbee, 2013)

In general, it is good practice to modularize your program small or large. Applications that benefit from modular programming are large-scale web applications. Large-scale web applications are complex systems. They need to handle a large and increasing number of users, higher loads of traffic, and an exponentially growing data pool. An example of a large-scale web application is an online retail platform, which needs to manage vast amounts of user and product data, process transactions securely, and scale dynamically (Struk, 2023). For large-scale web applications, modular programming combined with microservices architecture (a form of modulation that breaks up a program into separate services) is crucial for manageability, efficiency, scalability, and maintainability.

A popular framework for web applications is React, it utilizes the concept of components as modules.

“In React, you develop your applications by creating reusable components that you can think of as independent Lego blocks. These components are individual pieces of a final interface, which, when assembled, form the application’s entire user interface.” (Herbert, 2023)

“React's component-based architecture allows you to create modular and reusable code that can help you scale your web application as your business grows. This makes React a great choice for developing large-scale applications that require maintainability, scalability, and flexibility.” (Hutsulyak, 2023)

-Alex

References:

Busbee, K. L. (2013, January 10). *Programming Fundamentals - A Modular Structured Approach using C++.* Internet Archive.

Busbee, K. L., & Braunschweig, D. (2018, December 15). *Modular Programming*. Programming Fundamentals. https://press.rebus.community/programmingfundamentals/chapter/modular-programming/

Herbert, D. (2023, November 13). *What is react.js? uses, examples, & more*. HubSpot Blog. https://blog.hubspot.com/website/react-js

Hutsulyak, O. (2023, October 26). *Why use react for web development: 10 reasons to apply*. TechMagic. https://www.techmagic.co/blog/why-we-use-react-js-in-the-development/#:~:text=Large%2Dscale%20applications,maintainability%2C%20scalability%2C%20and%20flexibility.

Klein, B. (2022). *Intro to Python Tutorial*. Python Course. https://python-course.eu/python-tutorial/structuring-indentation.php

Macdonald, M. (2023). *Modular Programming: Definitions, benefits, and predictions*. Blog by Tiny. https://www.tiny.cloud/blog/modular-programming-principle/

Nguyen, P. (2018). *Programming Development Cycles*. California State University, Long Beach. https://home.csulb.edu/~pnguyen/cecs100/lecturenotes/Programming%20Development%20Cycles.docx

Struk, S. (2023, October 17). *Decoding the complexities of large-scale web application development*. Relevant Software. https://relevant.software/blog/guide-large-scale-web-application-development/