Dasar-dasar pemrograman

Ridwan Mahenra, S.Kom., M.C.S(AI)

Fakultas Teknik dan Ilmu Komputer
Universitas Teknokrat Indonesia

Ridwan Mahenra, S.Kom., M.C.S.(AI).

Expertise: Artificial Intelligence (AI) · Computer Vision · Robotics

Contact : ridwanmahenra@teknokrat.ac.id
: 082282910903 - **Chat only**

Education :

- **S1 Informatika** (**Universitas Teknokrat Indonesia**)
  2017-2021
- **S2 Kecerdasan Artifisial** (**Universitas Gadjah Mada**)
  2022-2023

License :

- *Expert* Machine Learning Foundations
  University of Washington
- *Expert* Generative Pre-trained Transformers (GPT)
  University of Glasgow
- *Professional* Transformer and BERT Modeling
  Google Cloud Certified

Award :

- **3rd Place** in the Thematic Robot Contest | KRI 2024
- **2nd Place** in the Humanoid Indonesian Football Robot Contest Division in the Running Competition Category | KRI 2020
- **3rd Place** in the Humanoid Indonesian Football Robot Contest Division in the Dribbling Competition Category | KRI 2020
- **1st Winner** of the Indonesian Flying Robot Contest, Vertical Take-off and Landing Division | KRTI 2019
- **3rd Place** in the Indonesian Rocket Cargo Competition (KOMURINDO 2019)

Course Description

This course provides the basics of computer programming theory. Students will learn the fundamental concepts of programming, data structures, and algorithms theoretically.

Learning Objectives:

- Understand basic programming concepts and related terminology.
- Understand the basic theory of data structures and algorithms.
- Able to analyze and design algorithms to solve computational problems.
- Understand the basic principles of object-oriented programming.
- Able to identify and correct logical errors in algorithms.

Weekly Topics and Details

Week 1: Introduction to Programming and its History
Week 2: Syntax and Program Structure Basics
Week 3: Program Flow Control
Week 4: Functions and Modularization
Week 5: Arrays and Basic Data Structures
Week 6: Strings and Data Manipulation
Week 7: File I/O (Input/Output)
Week 8: Midterm Exam (UTS)
Week 9: Recursion
Week 10: Introduction to Algorithms and Complexity
Week 11: Sorting and Searching
Week 12: Advanced Data Structures
Week 13: Introduction to Object Oriented Programming (OOP)
Week 14: Algorithm Design Principles
Week 15: Review and Discussion
Week 16: Final Semester Exam (UAS)

Teaching Methods

- Lecture and class discussion
- Discussion of case studies
- Individual or group assignments and presentations
- Analysis of algorithms and data structures through papers and discussions

Assessment:

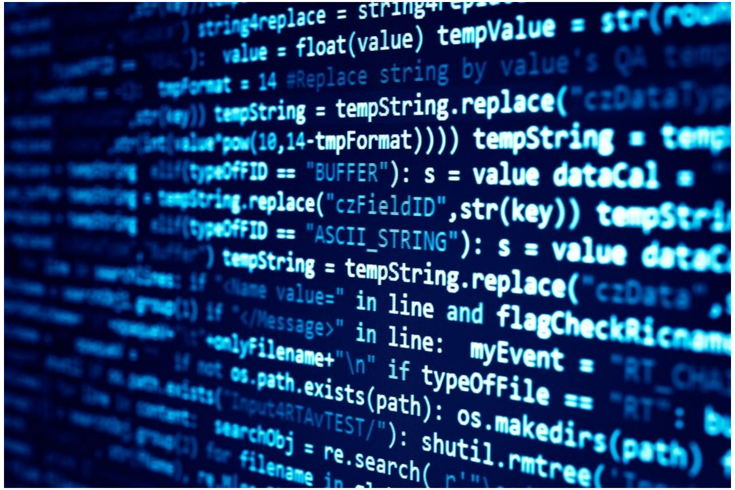
- Assignment and Presentation: 30% **Minimum attendance 80%*
- Quizzes and Practicum: 25
- Midterm Exam: 20%
- Final Semester Exam: 25%
-

Reference:
"Introduction to the Theory of Computation" by Michael Sipser
"Algorithm Design" by Jon Kleinberg and Éva Tardos

Introduction

Computer Programming: What is it and Why is it Important?
Computer programming is the art and science of writing instructions that can be executed by a computer to perform a specific task. These instructions are referred to as code or programs, and are written using a programming language specifically designed to communicate with computers.

Definition and Essence of Programming
Programming allows us to precisely control the behavior of a computer, providing step-by-step instructions that must be followed to achieve a desired result.



How does Programming Work?

1. **Writing Instructions (Code):**
   Programming begins with writing a set of instructions using a specific programming language.

2. **Compilation and Interpretation:**
   Once the code is written, it must be converted into a format that the computer can understand. This is done through two main methods:
   - Compilation
   - Interpretation

3. **Program Execution:**
   The computer will follow the instructions given in the code to perform a specific task, such as processing data, controlling hardware, or displaying information on the screen.

Why is Programming Important?

- **Task Automation**
- **Problem Solving**
- **Technology Innovation**
- **Skills Needed in the Future**

Early History of Programming

- **19th century**: Augusta Ada Lovelace, known as the first programmer, worked with Charles Babbage on analytical machines.
- **1940s**: Creation of the first computers such as ENIAC and Mark I. John von Neumann introduced the von Neumann architecture that became the basis for modern computers.

Programming Language Generation

- **First Generation (1940s):** Machine Language. Code written in binary and executed directly by the computer.
- **Second Generation (1950s):** Assembly Language. Using mnemonics for binary instructions, easier for humans to read.
- **Third Generation (1950s-1960s):** High-Level Language such as FORTRAN, COBOL, and Lisp. Closer to human language and required a compiler or interpreter.
- **Fourth Generation (1970s-1980s):** More abstract high-level languages such as SQL and MATLAB. Designed for specific uses and more productive.
- **Fifth Generation (1990s onwards):** Languages closer to logic programming and declarative programming such as Prolog and visual programming languages.

Programming Language Generation

Contemporary Developments:

- 1980s and 1990s: C, C++, Java, Python. Focus on efficiency, object-oriented programming, and widespread use in industry.
- 2000s onwards: Development of modern languages such as JavaScript, Swift, Kotlin, Go, Rust. Focus on security, performance, and development of mobile and web applications.

Definition of Programming

Programming is an art and science that involves the process of designing and building computer programs to solve problems or perform specific tasks.

What is Programming?

- **Creative and Technical Process:**
  Programming is a creative process that requires a programmer to design solutions in a logical and efficient manner.
- **Purpose of Programming:**
  - Problems Solving
  - Performing Specific Tasks

Steps in Programming

1. **Problem Analysis**
   The first step in programming is to understand and analyze the problem that needs to be solved.

2. **Algorithm Design**
   An algorithm is a series of logical steps that must be followed to solve a problem.

3. **Writing Code (Coding)**
   This process involves writing syntax and structures that conform to the rules of the programming language being used.

4. **Testing**
   Testing involves running the program with various inputs to check if the results are as expected.

5. **Debugging**
   Debugging is the process of looking for and fixing errors in the code.

6. **Maintenance**
   his includes updating the program to fix bugs, add new features, or adjust to changes in the environment or user needs.

Basic Elements of Programming

These elements are the foundation of almost all programming languages and are used to build complex programs.

1. **Variable**

   A variable is a storage place for data that can change during the course of the program.

   Declaration: Before being used, a variable must usually be declared. Declaring a variable involves giving it a name and, in some languages, specifying its data type.
   Initialization: Variables can be assigned an initial value at the time of declaration or at a later time during program execution.
   Modification: The value of a variable can be changed at any time in the program, which makes it flexible for use in various contexts.

   ```
   usia = 25
   nama = "Andi"

   usia = 26
   nama = "Budi"
   ```

2. Tipe Data

A data type determines the type of data that can be stored in a variable.

Types of Data Types:

- **Integer (int):** An integer, such as 1, 42, or -5.
- **Float (double, float):** A fractional or decimal number, such as 3.14, -0.001, or 2.71828.
- **String:** A sequence of characters, used to store text. Example: "Hello, world!" or "12345".
- **Boolean:** A logical value that can only be true or false.

```
age = 25 # Integer
height = 5.9 # Float
name = "Alice" # String
is_student = True # Boolean
```

```
int age = 25;          // Integer
double height = 5.9;     // Float
String name = "Alice";   // String
boolean isStudent = true; // Boolean
```

3. Operator

Operators are symbols used to perform operations on variables and values. Operators are used for various purposes such as arithmetic, comparison, and logic.

Arithmetic Operators:

Add (+) : Adds two values.
Subtract (-) : Subtracts one value from another.
Times (*) : Multiplies two values.
Divide (/) : Divides one value by another.
Modulus (%) : Produces the remainder of the division of two values.

```
a = 10
b = 5
c = a + b # Hasil: 15
d = a - b # Hasil: 5
e = a * b # Hasil: 50
f = a / b # Hasil: 2.0
g = a % b # Hasil: 0
```

Comparison Operator:

Equals (==) : Checks if two values are equal.
Not equal to (!=) : Checks if two values are not equal.
Greater than (>) : Checks whether the value on the left is greater than the value on the right.
Smaller than (<) : Checks whether the value on the left is smaller than the value on the right.
Greater than or equal to (>=) : Checks whether the value on the left is greater than or equal to the value on the right.
Smaller or equal to (<=) : Checks whether the value on the left is smaller or equal to the value on the right.

```
a = 10
b = 5
x = (a == b) # Hasil: False
z = (a > b) # Hasil: True
w = (a < b) # Hasil: False
u = (a >= b) # Hasil: True
v = (a <= b) # Hasil: False
```

Logical Operators:

And (&& or and) : Returns true if both operands are true.
Or (|| or or) : Returns true if either operand is true.
Not (! or not) : Reverses the logical value of the operand.

```
a = 10
b = 5
p = (a > 5 and b < 10) # Hasil: True
q = (a > 15 or b < 10) # Hasil: True
r = not (a == b) # Hasil: True
```