Aturan Praktikum Struktur Data

1. **Akun GitHub:** Setiap praktikan wajib memiliki akun GitHub yang aktif dan digunakan selama praktikum berlangsung.
2. **Invite Collaborator:** Setiap praktikan diwajibkan untuk menambahkan collaborator di setiap repository
    a. Asisten Praktikum: AndiniNH
    b. Asisten Praktikum : 4ldiputra
3. **Repository Praktikum:** Setiap praktikan diwajibkan untuk membuat satu repository di GitHub yang akan digunakan untuk seluruh tugas dan laporan praktikum. Repository ini harus diatur dengan rapi dan sesuai dengan instruksi yang akan diberikan di lampiran.
4. **Penamaan Folder:** Penamaan folder dalam repository akan dibahas secara rinci di lampiran. Praktikan wajib mengikuti aturan penamaan yang telah ditentukan.

Nomor	Pertemuan	Penamaan
1	Penganalan Bahasa C++ Bagian Pertama	01_Pengenalan_CPP_Bagian_1
2	Pengenalan Bahasa C++ Bagian Kedua	02_Pengenalan_CPP_Bagian_2
3	Abstract Data Type	03_Abstract_Data_Type
4	Single Linked List Bagian Pertama	04_Single_Linked_List_Bagian_1
5	Single Linked List Bagian Kedua	05_Single_Linked_List_Bagian_2
6	Double Linked List Bagian Pertama	06_Double_Linked_List_Bagian_1
7	Stack	07_Stack
8	Queue	08_Queue
9	Assessment Bagian Pertama	09_Assessment_Bagian_1
10	Tree Bagian Pertama	10_Tree_Bagian_1
11	Tree Bagian Kedua	11_Tree_Bagian_2
12	Asistensi Tugas Besar	12_Asistensi_Tugas_Besar
13	Multi Linked List	13_Multi_Linked_List
14	Graph	14_Graph
15	Assessment Bagian Kedua	15_Assessment_Bagian_2
16	Tugas Besar	16_Tugas_Besar

5. **Jam Praktikum:**
    - Jam masuk praktikum adalah **1 jam lebih lambat** dari jadwal yang tercantum. Sebagai contoh, jika jadwal praktikum adalah pukul 06.30 - 09.30, maka aturan praktikum akan diatur sebagai berikut:
        - **06.30 - 07.30**: Waktu ini digunakan untuk **Tugas Praktikum dan Laporan Praktikum** yang dilakukan di luar laboratorium.
        - **07.30 - 0G.30**: Sesi ini mencakup **tutorial, diskusi, dan kasus problem-solving**. Kegiatan ini berlangsung di dalam laboratorium dengan alokasi waktu sebagai berikut:
            - **60 menit pertama**: Tugas terbimbing.
            - **60 menit kedua**: Tugas mandiri.
6. **Pengumpulan Tugasn Pendahuluan:** Tugas Pendahuluan (TP) wajib dikumpulkan melalui GitHub sesuai dengan format berikut:

    nama_repo/nama_pertemuan/TP_Pertemuan_Ke.md

    Sebagai contoh:

    STD_Yudha_Islalmi_Sulistya_XXXXXXXX/01_Running_Modul/TP_01.md

7. **Pengecekan Tugas Pendahuluan:** Pengumpulan laporan praktikum akan diperiksa **1 hari sebelum praktikum selanjutnya** dimulai. Pastikan tugas telah diunggah tepat waktu untuk menghindari sanksi.

8. **Struktur Laporan Praktikum**

1. Cover :

LAPORAN PRAKTIKUM
Modul 3
“ABSTRACT DATA TYPE (ADT) ”

Disusun Oleh:
Reza Afiansyah Wibowo -2311104062
SE0702

Dosen :
WAHYU ANDI SAPUTRA

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2024

TUJUAN

Memahami konsep Abstract Data Type (ADT) dan penggunaannya dalam pemrograman.

DASAR TEORI

- Definisi/Spesifikasi Type dan Primitif/Header fungsi (.h)
  - Berisi spesifikasi tipe sesuai kaidah bahasa yang dipakai
  - Spesifikasi primitif (fungsi dan prosedur)

- Body/realisasi dari primitif (.c atau .cpp)
    - Berisi kode program dalam bahasa yang digunakan
    - Realisasi fungsi dan prosedur sebaiknya memanfaatkan selektor dan konstruktor

· Modul interface program utama (driver)

- Berisi program utama yang menggunakan ADT

KESIMPULAN

· ADT adalah konsep pemrograman yang terdiri dari TYPE (tipe data) dan sekumpulan PRIMITIF (operasi dasar) terhadap TYPE tersebut.
· ADT merupakan definisi statik yang dapat mencakup definisi invarian dari TYPE dan aksioma yang berlaku.
· Implementasi ADT biasanya terdiri dari tiga bagian utama: a. File header (.h) yang berisi definisi/spesifikasi type dan primitif b. File implementasi (.c atau .cpp) yang berisi realisasi dari primitif c. File program utama (driver) yang menggunakan ADT
· PRIMITIF dalam ADT dapat dikelompokkan menjadi beberapa jenis, termasuk konstruktor, selektor, prosedur pengubah nilai, validator, destruktor, fungsi input/output, operator relasional, operasi aritmatika, dan fungsi konversi.
· Penggunaan ADT memungkinkan pemisahan antara definisi dan implementasi, yang meningkatkan modularitas dan memudahkan pengembangan serta pemeliharaan kode.
· ADT dapat berisi definisi ADT lain, memungkinkan pembuatan struktur data yang lebih kompleks.
· Penerapan konsep ADT melibatkan pemisahan kode program menjadi beberapa file terpisah, yang meningkatkan organisasi dan struktur program.
· ADT membantu dalam abstraksi data, di mana detail implementasi disembunyikan dari pengguna, memungkinkan fokus pada penggunaan dan fungsionalitas daripada detail internal.