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 …
“Judul”

Disusun Oleh:
Reza Afiansyah Wibowo -2311104062
SE07B

Dosen :
WAHYU ANDI SAPUTRA

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2024

Tujuan

untuk memberikan pemahaman dasar tentang beberapa konsep penting dalam pemrograman C++, mulai dari struktur data sederhana seperti array, konsep pointer yang lebih kompleks, hingga pembuatan dan penggunaan fungsi. Materi ini juga menunjukkan beberapa praktik umum dalam penulisan kode C++, seperti penggunaan loop untuk iterasi dan pemisahan logika program ke dalam fungsi-fungsi.

Landasan Teori

1. Array:
    1. Definisi: Array adalah struktur data yang menyimpan kumpulan elemen dengan tipe data yang sama dalam lokasi memori yang berurutan.
    2. Teori: Array memungkinkan penyimpanan dan akses efisien ke sejumlah besar data menggunakan satu nama variabel dan indeks.
    3. Jenis: Array satu dimensi menyimpan list linear, sedangkan array multi-dimensi (seperti array 2D) menyimpan data dalam format tabel atau matriks.
2. Iterasi dan Loop:
    1. Konsep: Loop memungkinkan eksekusi berulang dari sekelompok pernyataan.
    2. Aplikasi: Berguna untuk memproses elemen-elemen dalam struktur data seperti array secara efisien.
3. Pointer:
    1. Definisi: Pointer adalah variabel yang menyimpan alamat memori dari variabel lain.
    2. Konsep: Pointer memungkinkan manipulasi memori langsung dan dapat meningkatkan efisiensi dalam beberapa operasi.
    3. Penggunaan: Berguna dalam alokasi memori dinamis, passing by reference, dan manipulasi struktur data kompleks.
4. Fungsi:
    1. Definisi: Fungsi adalah blok kode yang melakukan tugas spesifik dan dapat dipanggil dari bagian lain program.
    2. Jenis: Fungsi dengan nilai kembali (return value) dan fungsi void.
    3. Konsep: Mendukung modularitas, reusabilitas, dan abstraksi dalam pemrograman.
5. Paradigma Pemrograman:
    1. Prosedural: Pendekatan di mana program dibagi menjadi prosedur (fungsi) yang memanipulasi data.
    2. Awal Pemrograman Berorientasi Objek: Meskipun tidak eksplisit dalam kode, penggunaan fungsi dan struktur data mengarah ke konsep ini.
6. Manajemen Memori:
    1. Stack vs Heap: Variabel lokal disimpan di stack, sementara pointer dapat digunakan untuk mengakses memori di heap.
    2. Scope dan Lifetime: Konsep tentang di mana dan berapa lama variabel dapat diakses.
7. Input/Output:
    1. Stream I/O: Penggunaan objek seperti cout untuk output mendemonstrasikan konsep stream dalam C++.
8. Tipe Data:
    1. Tipe Data Primitif: Penggunaan int untuk bilangan bulat.
    2. Tipe Data Kompleks: Penggunaan string untuk teks.
9. Namespace:

1. Konsep: Namespace digunakan untuk mengelompokkan entitas yang berhubungan dan menghindari konflik penamaan.

10. Preprocessing dan Kompilasi:

    1. Header Files: Penggunaan #include menunjukkan bagaimana preprocessor bekerja dalam C++.
    2. Linking: Meskipun tidak eksplisit, penggunaan fungsi dari library standar melibatkan proses linking.

Landasan teori ini memberikan pemahaman dasar tentang bagaimana program C++ distruktur dan dieksekusi, serta konsep-konsep penting dalam pemrograman seperti penyimpanan data, kontrol alur program, modularitas, dan manajemen memori. Materi ini menyediakan fondasi yang kuat untuk memahami konsep-konsep pemrograman yang lebih lanjut.

Kesimpulan

materi ini memberikan pengenalan komprehensif terhadap konsep-konsep dasar pemrograman C++. Ini mencakup aspek penting dari manajemen data, kontrol program, dan strukturisasi kode. Pembelajaran ini meletakkan dasar yang kuat untuk pengembangan keterampilan pemrograman lebih lanjut, mempersiapkan pemahaman untuk konsep-konsep yang lebih kompleks dalam ilmu komputer dan pengembangan perangkat lunak. Materi ini juga membangun fondasi penting untuk transisi ke paradigma pemrograman yang lebih advanced seperti pemrograman berorientasi objek dan pemrograman fungsional di masa depan.