



UNIVERSITAS HASANUDDIN

ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA

MODUL I

Konsep Algoritma



Elly Warni
Ingrid Nurtanio
Amil Ahmad Ilham
Adnan
Anugrayani Bustamin

PRAKATA

Alhamdulillah, puji dan syukur kami panjatkan ke hadirat Allah SWT yang telah melimpahkan rahmat dan hidayah-Nya, sehingga modul ini dapat terselesaikan dengan baik. Shalawat serta salam senantiasa tercurah kepada junjungan kita, Nabi Muhammad SAW, beserta keluarga dan sahabat beliau yang telah membawa kita dari zaman kegelapan menuju terang benderang ilmu pengetahuan.

Modul ini disusun sebagai bagian dari rangkaian pembelajaran yang bertujuan untuk membantu mahasiswa memahami konsep dan pengertian algoritma. Modul ini merupakan modul pengantar materi konsep dasar dan pengertian algoritma, sehingga diharapkan mahasiswa memiliki pemahaman yang baik terhadap dasar konsep algoritma agar dapat membantu mahasiswa pada materi lanjutan.

Tidak dapat dipungkiri bahwa proses penyusunan modul ini tidak lepas dari dukungan berbagai pihak. Oleh karena itu, kami mengucapkan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan masukan dan bimbingan dalam proses penyusunan modul ini. Terima kasih kami sampaikan kepada rekan-rekan dosen dan staf yang telah memberikan kritik dan saran yang membangun, sehingga modul ini dapat terselesaikan dengan baik.

Kami menyadari bahwa modul ini masih memiliki keterbatasan dan kekurangan. Oleh karena itu, kami sangat mengharapkan masukan, kritik, dan saran dari para pembaca untuk penyempurnaan di masa yang akan datang. Semoga modul ini dapat memberikan manfaat dan menjadi bahan referensi yang berharga bagi para mahasiswa dalam memahami konsep algoritma dan pemrograman.



Akhir kata, semoga upaya ini mendapat ridha Allah SWT dan memberikan kontribusi positif bagi kemajuan ilmu pengetahuan. Amin.

Makassar, November 2024

Penulis

DAFTAR ISI

PRAKATA	i
DAFTAR ISI	iii
RPS MATA KULIAH ALGORITMA DAN STRUKTUR DATA.....	1
MODUL ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA.....	12
Pengantar.....	12
KEGIATAN BELAJAR 1.....	14
A. Deskripsi Singkat	14
B. Relevansi.....	15
C. Pembelajaran.....	16
D. Penilaian Kegiatan Belajar	23

RPS MATA KULIAH ALGORITMA DAN STRUKTUR DATA

	UNIVERSITAS HASANUDDIN FAKULTAS TEKNIK PROGRAM STUDI TEKNIK INFORMATIKA					Kode Dokumen
RENCANA PEMBELAJARAN SEMESTER						
MATA KULIAH (MK)	KODE	Rumpun MK	BOBOT (skt)		SEMESTER	Tgl Penyusunan
Algoritma Pemrograman dan Struktur Data	23D12110604	Teknik Informatika	T=2	P=2	2	19-02-2023
OTORISASI	Pengembang RPS	Koordinator RMK		Ketua PRODI		
	Dr. Ir. Ingrid Nurtanio., MT Dr. Amil Ahmad Ilham., ST., M.IT Adnan., ST., MT., PhD Anugrayani Bustamin., ST., MT Elly Warni, S.T., M.T	Dr. Ir. Ingrid Nurtanio., MT		Prof. Dr. Indrabayu., ST., M.T., M.Bus.Sys		
	CPL-PRODI yang dibebankan pada MK <i>Intended Learning Outcomes</i>					

Capaian Pembelajaran (CP)	CPL 1	Memiliki dasar pengetahuan Teknik Informatika yang meliputi teori dan konsep dasar dari Ilmu Komputer, Matematika dan Statistika, Algoritma dan Pemrograman, Rekayasa Perangkat Lunak, Manajemen Informasi dan Ketahanan Digital, serta pengetahuan tingkat lanjut pada bidang-bidang khusus Teknik Informatika, seperti Kecerdasan Buatan, Data Science, Jaringan Komputer, Komputasi Awan dan Internet of Things.
	CPL 3	Mampu mengaplikasikan pengetahuan bidang Teknik Informatika yang dipadankan dengan bidang ilmu lainnya untuk menganalisa dan mencari solusi dari berbagai masalah berbasis komputasi.
	CPL 4	Mampu mendesain, mengimplementasikan dan mengevaluasi solusi berbasis komputasi dengan mengaplikasikan ilmu Teknik Informatika dan dasar-dasar pembangunan perangkat lunak.
	Capaian Pembelajaran Mata Kuliah (CPMK)	
	CPMK	Setelah mengikuti Mata Kuliah Algoritma Pemrograman dan Struktur Data selama 1 (satu) semester, mahasiswa mampu menguasai dan menerapkan konsep teoritis bidang Informatika dan ilmu komputer secara dan mendalam serta memformulasikan penyelesaian masalah procedural dengan pemikiran logis dan sistematis secara mandiri dan terukur sebagai dasar pengembangan perangkat lunak.

	CPL ⇒ Sub-CPMK	
	CPL 1	Sub-CPMK 1 : Mahasiswa mampu mendefinisikan konsep dan pengertian Algoritma
	CPL 1	Sub-CPMK 2 : Mahasiswa mampu mendeksripsikan Dasar Algoritma
	CPL 3	Sub-CPMK 3 : Mahasiswa mampu mengimplementasikan Flowchart
	CPL 1	Sub-CPMK 4 : Mahasiswa mampu memahami bentuk pemilihan dan perulangan
	CPL 4	Sub-CPMK 5 : Mahasiswa mampu mengimplementasikan prosedur, fungsi dan pemrosesan teks
	CPL 1	Sub-CPMK 6 : Mahasiswa mampu mempraktekkan konsep array
	CPL 4	Sub-CPMK 7 : Mahasiswa mampu menetapkan ciri dari teknik pencarian
	CPL 4	Sub-CPMK 8 : Mahasiswa mampu menetapkan ciri dari teknik sorting dan menganalisis complexitas algoritma
	CPL 1	Sub-CPMK 9 : Mahasiswa mampu memahami struct dan konsep linked list
	CPL 1	Sub-CPMK 10 : Mahasiswa mampu memahami Antrian dan Stack
	CPL 1	Sub-CPMK 11 : Mahasiswa mampu memahami teknik graph
	CPL 3	Sub-CPMK 12 : Mahasiswa mampu mengimplementasi konsep Tree
Deskripsi Singkat MK	<p>Mata kuliah Algoritma Pemrograman dan Struktur Data merupakan mata kuliah yang menyajikan konsep pemrograman secara mendalam terkait algoritma dan struktur data. Muatan mata kuliah merupakan dasar dari ilmu matematika, algoritma dan logika yang menjadi syarat utama pada mata kuliah Pemrograman lanjut . Mata kuliah ini disajikan untuk mahasiswa semester dua di Program Studi Teknik Informatika, Penanggungjawab mata kuliah adalah Laboratorium Komputer . Metode Pembelajaran bauran, teori dan praktik</p>	
Bahan Kajian / Materi Pembelajaran	<ol style="list-style-type: none"> 1. Konsep Algoritma 2. Dasar Algoritma (Aturan Penulisan Header, deklarasi dan deskripsi) 3. Flowchart (Simbol, variabel, algoritma cabang, runtunan) 4. Pemilihan (Alir Kontrol dan Pengulangan) 5. Prosedur, Fungsi dan Pemrosesan Teks 6. Array – Larik 7. Pencarian (Binary Search) 	

	<ul style="list-style-type: none"> 8. Pencarian (Sequential Search) 9. Bubble dan Insertion Sort 10. Merge dan selection sort 11. Quick sort dan complexita algoritma 12. Struct dan Linked List 13. Antrian dan Stack 14. Graph berarah dan tidak berarah 15. Tree
--	---

Pustaka	Utama :	
	<ul style="list-style-type: none"> 1. Munir, Rinaldi dan Lidya, Leony. 2016. Algoritma dan Pemrograman dalam Bahasa Pascal, C dan C++ Edisi Keenam. Informatika : Bandung 2. Gozali, William dan Aji, Alham Fikri. 2014. Pemrograman Kompetitif Dasar. Ikatan Alumni Tim Olimpiade Komputer Indonesia. 3. Zakaria, Teddy M dan Prijono, Agus. 2005. Konsep dan Implementasii Struktur Data. Informatika : Bandung. 4. Niswar, M., Ilham, A., Zainuddin, Z., Adnan, A., P, A., Warni, E., Aswad, I., & Muslimin, Z. (2021). Sosialisasi Metode Berfikir Komputasional pada Pendidikan Dasar dan Menengah di Lingkup Sulawesi Selatan. URNAL EPAT eknologi erapan ntuk engabdian asyarakat, 4(1), 46-52. https://doi.org/10.25042/jurnal_tepat.v4i1.172 	
	Pendukung:	
	Tuliskan pustaka pendukung jika ada, sebagai pengayaan literasi	

Dosen Pengampu		1. Dr. Ir. Ingrid Nurtanio., MT 2. Dr. Amil Ahmad Ilham., ST., M.IT 3. Adnan., ST., MT., PhD 4. Anugrayani Bustamin., ST., MT 5. Elly Warni, S.T., M.T						
Matakuliah syarat		Dasar Pemrograman Komputer						
Pekan Ke-	Sub-CPMK (Kemampuan akhir tiap tahapan belajar)	Penilaian		Metode Pembelajaran, Penugasan Mahasiswa, [Estimasi Waktu]			Materi Pembelajaran [Pustaka]	Bobot Penilaian (%)
		Indikator	Kriteria & Bentuk	Luring (offline)	Daring (online)			
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	
1	Mahasiswa mampu mengetahui konsep dan pengertian Algoritma	Mengetahui konsep definisi, manfaat, mekanisme dan kontribusi Algoritma	Assignments	Lecture TM: 4 x 50 menit.	Lecture VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Konsep dan definisi algoritma 2. Mekanisme dan Kontribusi Algoritma	5%	

2	Mahasiswa mampu memahami Dasar Algoritma CPL 3- SubCPMK 2	Memahami Dasar Algoritma (tipe data, aturan penulisan, proses, instruksi dan aksi)	quizzes	Lecture, case study TM: 4 x 50 menit.	Lecture, case study VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Dasar Algoritma (tipe data, aturan penulisan, proses, instruksi, dan aksi) Pustaka : [1]	5%
3	Mahasiswa mampu mengenali Flowchart CPL 4- SubCPMK 3	Mampu mengenali Flowchart (simbol, variabel, algoritma cabang, dan runtunan)	Assignments	Lecture, small group discussion TM: 4 x 50 menit.	Lecture, small group discussion VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit	Flowchart (Simbol, variabel, algoritma cabang, runtunan) Pustaka : [1]	7%
4	Mahasiswa mampu memahami bentuk pemilihan (alih kontrol dan pengulangan)	Mampu memahami bentuk pemilihan (alih kontrol dan pengulangan)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses	Pemilihan Kontrol (Alir dan Pengulangan) Pustaka : [1]	7%

					/104D4224/inde x.php?id_session =13610		
5	Mahasiswa mampu mengimplementasikan prosedur, fungsi dan pemrosesan teks	Mampu mengimplementasikan pemanggilan prosedur, pemanggilan fungsi dan pemrosesan teks	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Pemanggilan Prosedur 2. Pemanggilan Fungsi 3. Pemrosesan Teks Pustaka : [1]	7%
6	Mahasiswa mampu mempraktekkan konsep array	Mampu mempraktekkan konsep array seperti menginisialisasi array, mengisi elemen, mencari nilai maksimum dan minimum	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Menginisialisasi array 2. Mengisi elemen array 3. Mencari nilai maksimum dan minimum Pustaka : [1][2]	7%

7	Mahasiswa mampu menetapkan ciri dari teknik pencarian	Mampu menetapkan ciri dari teknik pencarian ● Binary Search ● Sequential Search	Assignments quizzes	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Binary Search 2. Sequential Search Pustaka : [2][3]	7%
8	Evaluasi Tengah Semester / Ujian Tengah Semester						
9	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting ● Bubble Sort ● Insertion Sort	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Bubble Sort 2. Insertion Sort Pustaka : [2][3]	9%
	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting	Assignments	Lecture, practicum	Lecture, practicum VC: 4 x 50 menit.	1. Merge Sort Selection Sort Pustaka : [2][3]	9%

10		Merge Sort Selection Sort		TM: 4 x 50 menit.	PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610		
11	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting Quick Sort berserta complexitas Algoritmanya	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Quick Sort 2. Complexitas Algoritma Pustaka : [2][3]	9%
12	Mahasiswa mampu memahami struct dan konsep linked list	Mampu memahami struct dan konsep dan operasi dasar linked list (deklarasi simpul, penambahan)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit	1. Konsep Struct dan implementasinya 2. Deklarasi list 3. Penambahan simpul pada list awal, akhir dan tengah	7%

		(simpul pada linked list awal, akhir dan tengah)			https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Pustaka : [2][3]	
13	Mahasiswa mampu memahami Antrian dan Stack	Mampu memahami Antrian dan Stack dalam List	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Antrian dan Stack dalam list Pustaka : [2][3]	7%
14	Mahasiswa mampu memahami teknik graph	mampu memahami teknik graph (graph berarah dan tidak berarah menggunakan linked list)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Graph berarah Graph tidak berarah Pustaka : [2][3]	7%

					4224/index.php? id_session=1361 0		
15	Mahasiswa mampu mengimplementasi konsep Tree	Mampu mengimplementasi konsep Tree menggunakan linked list	Assignments quizzes	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Konsep tree menggunakan linked List Pustaka : [2][3]	7%
16	Evaluasi Akhir Semester / Ujian Akhir Semester						

MODUL ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA

MODUL III

FLOWCHART



Pengantar

Modul ini dirancang untuk membantu mahasiswa memahami konsep dasar dan pengertian algoritma, termasuk definisi, struktur, dan logika yang mendasari proses pemecahan masalah dalam pemrograman. Dalam rangka mencapai capaian pembelajaran mata kuliah (CPMK) yang telah ditetapkan, mahasiswa akan diarahkan untuk mengenali dan mengidentifikasi elemen-elemen penting dalam penyusunan algoritma.

Pada Sub-CPMK kali ini, mahasiswa diharapkan mampu memahami konsep algoritma secara menyeluruh serta mengetahui peran dan manfaatnya dalam merancang solusi yang efisien dan terstruktur. Kegiatan pembelajaran akan dilakukan melalui diskusi kelompok kecil dan pembelajaran melalui ceramah, serta didukung oleh platform pembelajaran daring.

Kegiatan pembelajaran ini akan mencakup,

- Tatap Muka (TM): 4 sesi, masing-masing 50 menit.
- Virtual Classroom (VC): 4 sesi, masing-masing 50 menit.
- Praktikum Terbimbing (PT): 4 sesi, masing-masing 60 menit.
- Bimbingan Mandiri (BM): 4 sesi, masing-masing 60 menit.

Sumber pembelajaran yang dapat diakses untuk mendukung pemahaman lebih lanjut tersedia di platform sikola, dengan pustaka referensi yang telah disediakan.

Setelah mengikuti modul ini, mahasiswa diharapkan dapat mencapai tingkat pemahaman yang cukup, setidaknya 80%, sebelum melanjutkan ke materi berikutnya.

KEGIATAN BELAJAR 1

KONSEP ALGORITMA

A. Deskripsi Singkat

Mahasiswa yang mempelajari konsep algoritma sering kali dihadapkan pada tantangan dalam memahami elemen-elemen dasar yang menjadi pondasi logika pemrograman, seperti struktur, variabel, dan alur logika. Pemahaman yang kuat mengenai konsep algoritma sangat penting karena algoritma merupakan dasar dari setiap proses pemrograman, di mana langkah-langkah sistematis dibentuk untuk menyelesaikan masalah secara efektif dan efisien.

Pada kegiatan belajar pertama ini, mahasiswa akan diperkenalkan pada konsep dasar algoritma dan cara-cara penerapannya dalam membangun proses pemrograman. Materi ini akan mencakup pengenalan elemen-elemen kunci dalam algoritma, seperti, struktur dasar runtunan (sequence) yang menyusun langkah-langkah berurutan, percabangan (branching) yang memungkinkan pemilihan kondisi, serta perulangan (looping) yang mengatur pengulangan proses. Selain itu, mahasiswa akan mempelajari alur logika yang digunakan dalam setiap algoritma untuk mencapai hasil akhir yang diinginkan.

Struktur dasar algoritma biasanya mengikuti alur yang logis dan sistematis, dimulai dari titik awal yang diikuti oleh langkah-langkah terencana hingga menghasilkan solusi atau hasil akhir. Dalam kegiatan belajar ini, mahasiswa juga akan diajak untuk mengenal berbagai pola dalam penyusunan algoritma, termasuk penggunaan cabang keputusan dan alur bercabang yang kerap ditemui dalam penyelesaian masalah dengan kondisi-kondisi tertentu. Dengan memahami elemen-elemen ini, mahasiswa dapat menyusun algoritma yang tidak hanya akurat tetapi

juga optimal, menghindari proses yang rumit dan berpotensi menimbulkan kesalahan.

Melalui kegiatan pembelajaran ini, mahasiswa diharapkan dapat memahami dan mengaplikasikan algoritma sebagai alat bantu dalam merancang solusi pemrograman yang sistematis dan terstruktur. Dengan kemampuan ini, mahasiswa dapat mengembangkan keterampilan berpikir logis yang akan sangat bermanfaat dalam memecahkan berbagai masalah pemrograman serta meningkatkan efektivitas dan efisiensi dalam pengembangan perangkat lunak.

B. Relevansi

Modul ini merupakan dasar awal yang sangat penting sebelum mahasiswa mempelajari materi-materi selanjutnya, khususnya dalam memahami dan menguasai konsep algoritma. Pada modul ini, mahasiswa akan diperkenalkan pada konsep dasar dan pengertian algoritma yang menjadi fondasi dalam perancangan logika pemrograman. Pemahaman yang diperoleh dalam modul ini akan memberikan landasan kuat bagi mahasiswa untuk dapat menyusun langkah-langkah logis yang sistematis, yang nantinya akan diimplementasikan baik dalam bentuk diagram alur (flowchart) di modul-modul berikutnya, kode semu (pseudocode), maupun dalam bentuk bahasa pemrograman.

Pemahaman algoritma di tahap awal ini memungkinkan mahasiswa untuk membangun kerangka berpikir yang logis dan terstruktur dalam menyusun solusi pemrograman. Mereka akan mempelajari komponen-komponen mendasar seperti variabel, urutan proses, dan keputusan dalam logika algoritma, yang merupakan elemen penting untuk dapat menggambarkan dan memahami alur proses secara mendalam.

Dalam rangkaian modul ini, pemahaman konsep algoritma sangat relevan terhadap pencapaian Sub-Capaian Pembelajaran Mata Kuliah

(Sub-CPMK), karena memberikan dasar analitis bagi mahasiswa untuk dapat merancang algoritma secara logis dan efisien. Dengan pemahaman ini, mahasiswa akan lebih siap untuk mengembangkan keterampilan teknis yang lebih tinggi dalam memvisualisasikan dan mengimplementasikan algoritma di modul-modul selanjutnya.

C. Pembelajaran

Capaian pembelajaran yang diharapkan pada kegiatan belajar yang pertama ini adalah peserta kuliah akan mampu mengetahui konsep dan pengertian algoritma. Untuk mencapai kompetensi lulusan ini, peserta kuliah diharapkan membaca dan menyelesaikan Kegiatan Belajar 1 ini. Untuk mencapai capaian pembelajaran pada Kegiatan Belajar 1, peserta kuliah harus mengikuti urutan kegiatan pembelajaran yang telah disiapkan dalam modul ini.

1. Uraian

1.1 Pendahuluan

Algoritma dalam pemrograman adalah kumpulan langkah-langkah terstruktur yang digunakan untuk menyelesaikan masalah atau mencapai tujuan tertentu. Algoritma memang dirancang secara sistematis dan logis supaya mudah diikuti maupun diimplementasikan oleh komputer ataupun sistem pemrosesan sebuah data lainnya. Dalam artian sederhana, algoritma pemrograman dasar logika dalam pembuatan program komputer supaya bisa berjalan sesuai perintah. Algoritma ini bertujuan untuk membantu memecahkan masalah secara lebih sistematis dan terstruktur, sehingga solusi dapat ditemukan dengan mudah. Untuk fungsi utama dari algoritma sendiri adalah membantu para programmer dalam merancang maupun menulis kode secara efektif, efisien dan bisa dipahami dengan mudah.

Dalam dunia pemrograman, algoritma berperan untuk merancang berbagai jenis program. Algoritma juga dapat diekspresikan melalui bahasa normal, flowchart, pseudocode, dan berbagai bahasa pemrograman seperti C, C++, Java, dll. Dalam pemrograman, algoritma bekerja dengan mengandalkan pada tiga aspek utama dalam membangun programnya yaitu input, proses, output. Dimana program dengan algoritma yang digunakan akan memproses inputan yang diberikan sehingga dapat memberikan output sesuai yang diinginkan.

Menurut Donald E.Knuth sebuah algoritma harus memenuhi persyaratan :

- Finiteness. Algoritma harus berakhir (terminate) setelah melakukan sejumlah langkah proses.
- Definiteness. Setiap langkah algoritma harus didefinisikan dengan tepat dan tidak menimbulkan makna ganda(ambiguous).
- Input. Setiap algoritma memerlukan data sebagai masukan untuk diolah.
- Output. Setiap algoritma memberikan satu atau lebih hasil keluaran.
- Effectiveness. Langkah-langkah algoritma dikerjakan dalam batas waktu yang wajar.

1.2 Proses Pada Algoritma

Langkah-langkah yang membentuk suatu algoritma dapat dibagi menjadi tiga kelompok proses, yaitu :

1. Sequence process.

Sederetan instruksi dijalankan secara berurutan dari awal hingga akhir.

2. Selection process.

Instruksi atau sederetan instruksi dijalankan jika kondisi tertentu terpenuhi. Contohnya adalah siswa dinyatakan lulus mata kuliah jika nilainya minimal 60.

3. Iteration process.

Instruksi atau sederetan instruksi dijalankan secara berulang jika kondisi tertentu terpenuhi. Contohnya adalah siswa harus mengambil mata kuliah selama nilainya di bawah 60

1.3 Jenis Algoritma

Ada berbagai jenis algoritma berdasarkan tujuan dan cara kerjanya, beberapa diantaranya yaitu:

1. Searching Algorithm (Algoritma Pencarian)

Algoritma pencarian digunakan untuk menemukan posisi atau nilai tertentu dalam kumpulan data. Algoritma pencarian menjadi penting terutama ketika kita bekerja dengan data dalam jumlah besar.

2. Sorting Algorithm (Algoritma Pengurutan)

Algoritma pengurutan digunakan untuk menyusun data dalam urutan tertentu, baik itu ascending (menaik) maupun descending (menurun).

3. Graph Algorithm (Algoritma Graf)

Algoritma graf dirancang untuk menyelesaikan masalah yang melibatkan graf, seperti jalur terpendek, pengelompokan, dan jaringan.

4. Brute Force Algorithm



Algoritma brute force mencoba semua kemungkinan solusi untuk menemukan solusi yang benar.

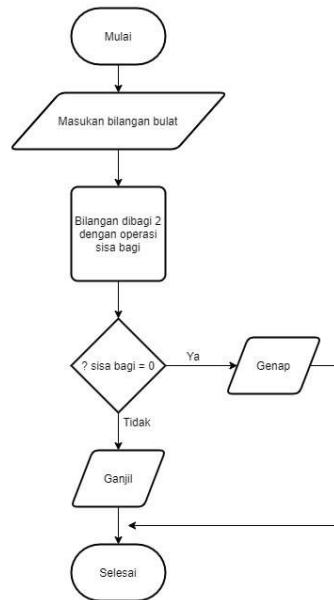
1.4 Efisiensi Algoritma

Setiap algoritma memiliki cara dan tujuannya masing-masing dalam memberikan solusi pada suatu masalah tertentu. Tentu saja tiap algoritma memiliki keefesienan yang berbeda dalam mengeksekusi masalah yang diberikan. Efisiensi algoritma dinilai berdasarkan seberapa cepat algoritma berjalan dan berapa banyak memori yang dibutuhkan untuk menyelesaikan masalah, terutama saat ukuran input bertambah. Dua aspek utama yang diperhatikan adalah kompleksitas waktu (Time Complexity) dan kompleksitas ruang (Space Complexity). Kompleksitas waktu mengukur kecepatan algoritma dengan menggunakan notasi Big-O untuk menunjukkan seberapa cepat waktu eksekusi meningkat sesuai pertambahan ukuran input. Sementara itu, kompleksitas ruang mengukur jumlah memori yang dibutuhkan oleh algoritma selama proses eksekusi.

1.5 Implementasi Algoritma

Algoritma dapat diterapkan melalui berbagai metode, antara lain:

- **Flowchart:** Diagram visual untuk memetakan urutan langkah dalam suatu proses, sangat berguna untuk memahami alur logika.



- **Pseudocode:** Deskripsi berbasis teks yang menyerupai kode, tetapi tidak menggunakan sintaks bahasa pemrograman tertentu.

```

01| ALGORITMA Menentukan_terbesar_dari_3_bilangan
02| Deklarasi:
03|   a,b,c, terbesar : integer
04|
05| Deskripsi:
06|   Read(a,b,c)
07|   If (a>b) and (a>c) then
08|     Terbesar ← a
09|   Else
10|     If b>c then
11|       Terbesar ← b
12|     Else
13|       Terbesar ← c
14|     Endif
15|   Endif
16|   Write(terbesar)
  
```

- **Bahasa Pemrograman:** Setelah algoritma dirancang, ia dapat diimplementasikan ke dalam bahasa pemrograman seperti Python, Java, atau C++.

Source Code Program dalam C++

```

#include <bits/stdc++.h>
using namespace std;
  
```

```
#define ll long long
#define mp make_pair
const int limit = 1e6 + 5;

vector<ll> valuePrime()
{
    vector<ll> primeVal;
    vector<bool> isPrime(limit, true);

    for (int i = 2; i * i <= limit; i++)
    {
        if (isPrime[i])
        {
            for (int j = i * i; j <= limit; j += i)
            {
                isPrime[j] = false;
            }
        }
    }

    for (int i = 2; i <= limit; i++)
    {
        if (isPrime[i])
        {
            primeVal.push_back(i);
        }
    }
}
```

```
        }

    }

    return primeVal;
}

int main()
{
    int t;
    cin >> t;
    vector<int> index(t);
    vector<ll> ans = valuePrime();

    for (int i = 0; i < t; i++)
    {
        cin >> index[i];
    }

    for (int i = 0; i < t; i++)
    {
        cout << ans[index[i] - 1] << endl;
    }

    return 0;
}
```

2. Rangkuman

Algoritma dalam pemrograman adalah serangkaian langkah-langkah terstruktur untuk menyelesaikan masalah atau mencapai tujuan tertentu dengan logis dan sistematis, sehingga memudahkan programmer dalam menulis kode yang efektif dan efisien. Algoritma dapat dinyatakan dalam bahasa biasa, flowchart, pseudocode, atau berbagai bahasa pemrograman seperti C++ dan Java, dan berfokus pada tiga aspek utama: input, proses, dan output. Menurut Donald E. Knuth, algoritma harus memenuhi beberapa syarat: finiteness (berakhir setelah beberapa langkah), definiteness (jelas dan tidak ambigu), input (memerlukan data untuk diolah), output (menghasilkan satu atau lebih keluaran), dan effectiveness (dijalankan dalam batas waktu wajar). Terdapat tiga jenis proses utama dalam algoritma, yaitu sequence (instruksi berurutan dari awal hingga akhir), selection (instruksi dijalankan jika kondisi tertentu terpenuhi), dan iteration (instruksi berulang saat kondisi tertentu terpenuhi).

3. Pustaka

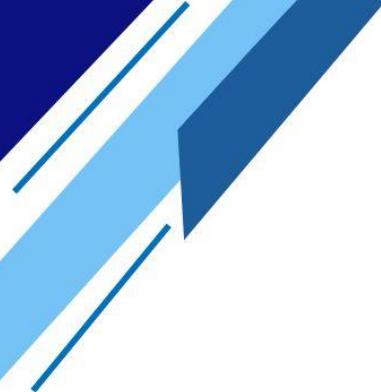
- [1] Huda, C. (2017). Pengenalan Algoritma.

D. Penilaian Kegiatan Belajar

1. Penilaian

1.1 Latihan

- 1) Cari tahu mengenai apa saja jenis algoritma yang sering digunakan pada pemrograman dan bagaimana cara kerjanya!

- 
- 2) Mengapa penting untuk memilih algoritma yang tepat dalam pemecahan masalah? Berikan contoh kasus di mana pemilihan algoritma berpengaruh pada hasil!

1.2 Tantangan

- 1) Seperti yang kita ketahui bahwa algoritma brute force bekerja dengan mencoba semua kemungkinan pada program. Apa perbedaan antara algoritma brute force dengan algoritma lain dari segi kompleksitas ruang dan waktu? Dalam situasi seperti apa metode brute force masih relevan digunakan?

2. Umpulan Balik dan Tindak Lanjut

1) Jenis Algoritma dan Cara Kerjanya

Algoritma yang sering digunakan dalam pemrograman mencakup algoritma *sorting* (pengurutan), *searching* (pencarian), dan *graph*. Algoritma *sorting* mengatur elemen dalam urutan tertentu, biasanya berdasarkan nilai. Contoh algoritma *sorting* seperti *Merge Sort* dan *Quick Sort* menggunakan pendekatan *divide and conquer* untuk membagi data menjadi bagian-bagian lebih kecil, lalu menyusunnya kembali secara terurut dengan efisiensi yang baik (kompleksitas $O(n \log n)$). *Searching* adalah algoritma yang digunakan untuk menemukan posisi atau keberadaan elemen dalam data. Algoritma seperti *binary search* bekerja dalam waktu $O(\log n)$ pada data yang sudah diurutkan, membagi data menjadi dua pada setiap langkah hingga menemukan elemen yang dicari. Ini jauh lebih efisien dibandingkan pencarian linier, yang memiliki kompleksitas $O(n)$. Algoritma *graph* digunakan untuk memecahkan masalah yang melibatkan hubungan antar objek, seperti pencarian jalur terpendek. Algoritma *Dijkstra*, misalnya, membantu menemukan jalur terpendek dalam graf berbobot





positif dengan mengeksplorasi semua simpul yang terhubung secara sistematis. Algoritma graf lain, seperti *Depth-First Search (DFS)* dan *Breadth-First Search (BFS)*, digunakan untuk penjelajahan graf, yang berguna dalam masalah seperti deteksi siklus atau komponen terhubung pada graf.

2) Memilih Algoritma yang Tepat

Memilih algoritma yang tepat sangat penting karena memengaruhi efisiensi waktu dan memori yang dibutuhkan untuk menyelesaikan suatu masalah. Algoritma yang tepat dapat mengurangi waktu komputasi secara signifikan, terutama pada data skala besar. Sebagai contoh, untuk mencari elemen dalam data terurut, algoritma pencarian biner jauh lebih efisien dibandingkan pencarian linier. Dalam kasus ini, pencarian biner bekerja dalam kompleksitas waktu logaritmik $O(\log n)$, sedangkan pencarian linier membutuhkan $O(n)$. Dengan memilih algoritma yang lebih optimal, kinerja aplikasi atau sistem bisa menjadi jauh lebih cepat dan menghemat sumber daya.



ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA

MODUL II

*Dasar
Algoritma*



Elly Warni
Ingrid Nurtanio
Amil Ahmad Ilham
Adnan
Anugrayani Bustamin

PRAKATA

Alhamdulillah, puji dan syukur kami panjatkan ke hadirat Allah SWT yang telah melimpahkan rahmat dan hidayah-Nya, sehingga modul ini dapat terselesaikan dengan baik. Shalawat serta salam senantiasa tercurah kepada junjungan kita, Nabi Muhammad SAW, beserta keluarga dan sahabat beliau yang telah membawa kita dari zaman kegelapan menuju terang benderang ilmu pengetahuan.

Modul ini disusun sebagai bagian dari rangkaian pembelajaran yang bertujuan untuk membantu mahasiswa memahami dasar algoritma. Modul ini merupakan modul lanjutan dari modul sebelumnya yang membahas konsep dan pengertian algoritma, sehingga pada modul ini, diharapkan mahasiswa memiliki pemahaman lebih dalam terhadap konsep algoritma dan penerapannya.

Tidak dapat dipungkiri bahwa proses penyusunan modul ini tidak lepas dari dukungan berbagai pihak. Oleh karena itu, kami mengucapkan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan masukan dan bimbingan dalam proses penyusunan modul ini. Terima kasih kami sampaikan kepada rekan-rekan dosen dan staf yang telah memberikan kritik dan saran yang membangun, sehingga modul ini dapat terselesaikan dengan baik.

Kami menyadari bahwa modul ini masih memiliki keterbatasan dan kekurangan. Oleh karena itu, kami sangat mengharapkan masukan, kritik, dan saran dari para pembaca untuk penyempurnaan di masa yang akan datang. Semoga modul ini dapat memberikan manfaat dan menjadi bahan referensi yang berharga bagi para mahasiswa dalam memahami konsep algoritma dan pemrograman.



Akhir kata, semoga upaya ini mendapat ridha Allah SWT dan memberikan kontribusi positif bagi kemajuan ilmu pengetahuan. Amin.

Makassar, November 2024

Penulis

DAFTAR ISI

PRAKATA	i
DAFTAR ISI	iii
RPS MATA KULIAH ALGORITMA DAN STRUKTUR DATA.....	1
MODUL ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA.....	12
Pengantar.....	12
KEGIATAN BELAJAR 1.....	14
A. Deskripsi Singkat	14
B. Relevansi.....	14
C. Pembelajaran.....	15
D. Penilaian Kegiatan Belajar	19

RPS MATA KULIAH ALGORITMA DAN STRUKTUR DATA

	UNIVERSITAS HASANUDDIN FAKULTAS TEKNIK PROGRAM STUDI TEKNIK INFORMATIKA					Kode Dokumen
RENCANA PEMBELAJARAN SEMESTER						
MATA KULIAH (MK)	KODE	Rumpun MK	BOBOT (skt)		SEMESTER	Tgl Penyusunan
Algoritma Pemrograman dan Struktur Data	23D12110604	Teknik Informatika	T=2	P=2	2	19-02-2023
OTORISASI	Pengembang RPS	Koordinator RMK		Ketua PRODI		
	Dr. Ir. Ingrid Nurtanio., MT Dr. Amil Ahmad Ilham., ST., M.IT Adnan., ST., MT., PhD Anugrayani Bustamin., ST., MT Elly Warni, S.T., M.T	Dr. Ir. Ingrid Nurtanio., MT		Prof. Dr. Indrabayu., ST., M.T., M.Bus.Sys		
	CPL-PRODI yang dibebankan pada MK <i>Intended Learning Outcomes</i>					

Capaian Pembelajaran (CP)	CPL 1	Memiliki dasar pengetahuan Teknik Informatika yang meliputi teori dan konsep dasar dari Ilmu Komputer, Matematika dan Statistika, Algoritma dan Pemrograman, Rekayasa Perangkat Lunak, Manajemen Informasi dan Ketahanan Digital, serta pengetahuan tingkat lanjut pada bidang-bidang khusus Teknik Informatika, seperti Kecerdasan Buatan, Data Science, Jaringan Komputer, Komputasi Awan dan Internet of Things.
	CPL 3	Mampu mengaplikasikan pengetahuan bidang Teknik Informatika yang dipadankan dengan bidang ilmu lainnya untuk menganalisa dan mencari solusi dari berbagai masalah berbasis komputasi.
	CPL 4	Mampu mendesain, mengimplementasikan dan mengevaluasi solusi berbasis komputasi dengan mengaplikasikan ilmu Teknik Informatika dan dasar-dasar pembangunan perangkat lunak.
	Capaian Pembelajaran Mata Kuliah (CPMK)	
	CPMK	Setelah mengikuti Mata Kuliah Algoritma Pemrograman dan Struktur Data selama 1 (satu) semester, mahasiswa mampu menguasai dan menerapkan konsep teoritis bidang Informatika dan ilmu komputer secara dan mendalam serta memformulasikan penyelesaian masalah procedural dengan pemikiran logis dan sistematis secara mandiri dan terukur sebagai dasar pengembangan perangkat lunak.

	CPL ⇒ Sub-CPMK	
	CPL 1	Sub-CPMK 1 : Mahasiswa mampu mendefinisikan konsep dan pengertian Algoritma
	CPL 1	Sub-CPMK 2 : Mahasiswa mampu mendeksripsikan Dasar Algoritma
	CPL 3	Sub-CPMK 3 : Mahasiswa mampu mengimplementasikan Flowchart
	CPL 1	Sub-CPMK 4 : Mahasiswa mampu memahami bentuk pemilihan dan perulangan
	CPL 4	Sub-CPMK 5 : Mahasiswa mampu mengimplementasikan prosedur, fungsi dan pemrosesan teks
	CPL 1	Sub-CPMK 6 : Mahasiswa mampu mempraktekkan konsep array
	CPL 4	Sub-CPMK 7 : Mahasiswa mampu menetapkan ciri dari teknik pencarian
	CPL 4	Sub-CPMK 8 : Mahasiswa mampu menetapkan ciri dari teknik sorting dan menganalisis complexitas algoritma
	CPL 1	Sub-CPMK 9 : Mahasiswa mampu memahami struct dan konsep linked list
	CPL 1	Sub-CPMK 10 : Mahasiswa mampu memahami Antrian dan Stack
	CPL 1	Sub-CPMK 11 : Mahasiswa mampu memahami teknik graph
	CPL 3	Sub-CPMK 12 : Mahasiswa mampu mengimplementasi konsep Tree
Deskripsi Singkat MK	<p>Mata kuliah Algoritma Pemrograman dan Struktur Data merupakan mata kuliah yang menyajikan konsep pemrograman secara mendalam terkait algoritma dan struktur data. Muatan mata kuliah merupakan dasar dari ilmu matematika, algoritma dan logika yang menjadi syarat utama pada mata kuliah Pemrograman lanjut . Mata kuliah ini disajikan untuk mahasiswa semester dua di Program Studi Teknik Informatika, Penanggungjawab mata kuliah adalah Laboratorium Komputer . Metode Pembelajaran bauran, teori dan praktik</p>	
Bahan Kajian / Materi Pembelajaran	<ol style="list-style-type: none"> 1. Konsep Algoritma 2. Dasar Algoritma (Aturan Penulisan Header, deklarasi dan deskripsi) 3. Flowchart (Simbol, variabel, algoritma cabang, runtunan) 4. Pemilihan (Alir Kontrol dan Pengulangan) 5. Prosedur, Fungsi dan Pemrosesan Teks 6. Array – Larik 7. Pencarian (Binary Search) 	

	<ul style="list-style-type: none"> 8. Pencarian (Sequential Search) 9. Bubble dan Insertion Sort 10. Merge dan selection sort 11. Quick sort dan complexita algoritma 12. Struct dan Linked List 13. Antrian dan Stack 14. Graph berarah dan tidak berarah 15. Tree
--	---

Pustaka	Utama :	
	<ul style="list-style-type: none"> 1. Munir, Rinaldi dan Lidya, Leony. 2016. Algoritma dan Pemrograman dalam Bahasa Pascal, C dan C++ Edisi Keenam. Informatika : Bandung 2. Gozali, William dan Aji, Alham Fikri. 2014. Pemrograman Kompetitif Dasar. Ikatan Alumni Tim Olimpiade Komputer Indonesia. 3. Zakaria, Teddy M dan Prijono, Agus. 2005. Konsep dan Implementasii Struktur Data. Informatika : Bandung. 4. Niswar, M., Ilham, A., Zainuddin, Z., Adnan, A., P, A., Warni, E., Aswad, I., & Muslimin, Z. (2021). Sosialisasi Metode Berfikir Komputasional pada Pendidikan Dasar dan Menengah di Lingkup Sulawesi Selatan. URNAL EPAT eknologi erapan ntuk engabdian asyarakat, 4(1), 46-52. https://doi.org/10.25042/jurnal_tepat.v4i1.172 	
	Pendukung:	
	Tuliskan pustaka pendukung jika ada, sebagai pengayaan literasi	

Dosen Pengampu		1. Dr. Ir. Ingrid Nurtanio., MT 2. Dr. Amil Ahmad Ilham., ST., M.IT 3. Adnan., ST., MT., PhD 4. Anugrayani Bustamin., ST., MT 5. Elly Warni, S.T., M.T						
Matakuliah syarat		Dasar Pemrograman Komputer						
Pekan Ke-	Sub-CPMK (Kemampuan akhir tiap tahapan belajar)	Penilaian		Metode Pembelajaran, Penugasan Mahasiswa, [Estimasi Waktu]			Materi Pembelajaran [Pustaka]	Bobot Penilaian (%)
		Indikator	Kriteria & Bentuk	Luring (offline)	Daring (online)			
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	
1	Mahasiswa mampu mengetahui konsep dan pengertian Algoritma	Mengetahui konsep definisi, manfaat, mekanisme dan kontribusi Algoritma	Assignments	Lecture TM: 4 x 50 menit.	Lecture VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Konsep dan definisi algoritma 2. Mekanisme dan Kontribusi Algoritma	5%	

2	Mahasiswa mampu memahami Dasar Algoritma CPL 3- SubCPMK 2	Memahami Dasar Algoritma (tipe data, aturan penulisan, proses, instruksi dan aksi)	quizzes	Lecture, case study TM: 4 x 50 menit.	Lecture, case study VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Dasar Algoritma (tipe data, aturan penulisan, proses, instruksi, dan aksi) Pustaka : [1]	5%
3	Mahasiswa mampu mengenali Flowchart CPL 4- SubCPMK 3	Mampu mengenali Flowchart (simbol, variabel, algoritma cabang, dan runtunan)	Assignments	Lecture, small group discussion TM: 4 x 50 menit.	Lecture, small group discussion VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit	Flowchart (Simbol, variabel, algoritma cabang, runtunan) Pustaka : [1]	7%
4	Mahasiswa mampu memahami bentuk pemilihan (alih kontrol dan pengulangan)	Mampu memahami bentuk pemilihan (alih kontrol dan pengulangan)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses	Pemilihan Kontrol (Alir dan Pengulangan) Pustaka : [1]	7%

					/104D4224/inde x.php?id_session =13610		
5	Mahasiswa mampu mengimplementasikan prosedur, fungsi dan pemrosesan teks	Mampu mengimplementasikan pemanggilan prosedur, pemanggilan fungsi dan pemrosesan teks	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Pemanggilan Prosedur 2. Pemanggilan Fungsi 3. Pemrosesan Teks Pustaka : [1]	7%
6	Mahasiswa mampu mempraktekkan konsep array	Mampu mempraktekkan konsep array seperti menginisialisasi array, mengisi elemen, mencari nilai maksimum dan minimum	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Menginisialisasi array 2. Mengisi elemen array 3. Mencari nilai maksimum dan minimum Pustaka : [1][2]	7%

7	Mahasiswa mampu menetapkan ciri dari teknik pencarian	Mampu menetapkan ciri dari teknik pencarian • Binary Search • Sequential Search	Assignments quizzes	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Binary Search 2. Sequential Search Pustaka : [2][3]	7%
8	Evaluasi Tengah Semester / Ujian Tengah Semester						
9	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting • Bubble Sort • Insertion Sort	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Bubble Sort 2. Insertion Sort Pustaka : [2][3]	9%
	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting	Assignments	Lecture, practicum	Lecture, practicum VC: 4 x 50 menit.	1. Merge Sort Selection Sort Pustaka : [2][3]	9%

10		Merge Sort Selection Sort		TM: 4 x 50 menit.	PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610		
11	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting Quick Sort berserta complexitas Algoritmanya	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Quick Sort 2. Complexitas Algoritma Pustaka : [2][3]	9%
12	Mahasiswa mampu memahami struct dan konsep linked list	Mampu memahami struct dan konsep dan operasi dasar linked list (deklarasi simpul, penambahan)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit	1. Konsep Struct dan implementasinya 2. Deklarasi list 3. Penambahan simpul pada list awal, akhir dan tengah	7%

		(simpul pada linked list awal, akhir dan tengah)			https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Pustaka : [2][3]	
13	Mahasiswa mampu memahami Antrian dan Stack	Mampu memahami Antrian dan Stack dalam List	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Antrian dan Stack dalam list Pustaka : [2][3]	7%
14	Mahasiswa mampu memahami teknik graph	mampu memahami teknik graph (graph berarah dan tidak berarah menggunakan linked list)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Graph berarah Graph tidak berarah Pustaka : [2][3]	7%

					4224/index.php? id_session=1361 0		
15	Mahasiswa mampu mengimplementasi konsep Tree	Mampu mengimplementasi konsep Tree menggunakan linked list	Assignments quizzes	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Konsep tree menggunakan linked List Pustaka : [2][3]	7%
16	Evaluasi Akhir Semester / Ujian Akhir Semester						

MODUL ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA

MODUL II

DASAR ALGORITMA



Pengantar

Modul ini dirancang untuk membantu mahasiswa memahami konsep dasar algoritma, mulai dari definisi, tujuan, hingga penyusunan solusi logis yang efisien dalam pemrograman. Sebagai langkah awal untuk mencapai capaian pembelajaran mata kuliah (CPMK) yang telah ditetapkan, mahasiswa akan mempelajari konsep algoritma dan cara mengidentifikasi elemen-elemen kunci yang membentuknya, dengan penekanan pada penggunaan pseudocode sebagai bentuk implementasinya.

Pada Sub-CPMK ini, mahasiswa diharapkan mampu memahami dan menerapkan pseudocode sebagai cara sederhana dan terstruktur untuk menuliskan algoritma, tanpa perlu langsung menggunakan bahasa pemrograman. Dengan pseudocode, mahasiswa akan belajar menyusun langkah-langkah penyelesaian masalah dalam bentuk yang mudah dipahami dan diterjemahkan ke dalam bahasa pemrograman di tahap selanjutnya. Kegiatan pembelajaran akan dilakukan melalui diskusi kelompok kecil dan pembelajaran, serta didukung oleh platform pembelajaran daring.

Kegiatan pembelajaran ini akan mencakup,

- Tatap Muka (TM): 4 sesi, masing-masing 50 menit.
- Virtual Classroom (VC): 4 sesi, masing-masing 50 menit.

- Praktikum Terbimbing (PT): 4 sesi, masing-masing 60 menit.
- Bimbingan Mandiri (BM): 4 sesi, masing-masing 60 menit.

Sumber pembelajaran yang dapat diakses untuk mendukung pemahaman lebih lanjut tersedia di platform sikola, dengan pustaka referensi yang telah disediakan.

Setelah mengikuti modul ini, mahasiswa diharapkan dapat mencapai tingkat pemahaman yang cukup, setidaknya 80%, sebelum melanjutkan ke materi berikutnya.

KEGIATAN BELAJAR 1

DASAR ALGORITMA

A. Deskripsi Singkat

Materi ini dirancang untuk membantu mahasiswa memahami dasar-dasar algoritma, khususnya dalam mengenali dan menerapkan tipe data, aturan penulisan, proses, instruksi, dan aksi menggunakan pseudocode. Pemahaman akan dasar-dasar ini sangat penting, mengingat algoritma merupakan inti dari proses pemrograman. Dalam pemrograman, algoritma menyediakan langkah-langkah sistematis yang dibutuhkan untuk memecahkan masalah secara efektif dan efisien.

Pada kegiatan belajar pertama ini, mahasiswa akan diperkenalkan pada penggunaan pseudocode untuk menyusun algoritma yang akurat dan efisien. Dengan menguasai konsep ini, mahasiswa akan memiliki pondasi logis yang kuat dalam merancang solusi pemrograman, meningkatkan keterampilan berpikir sistematis, serta mengembangkan cara-cara pemecahan masalah yang terstruktur. Materi ini akan mencakup aturan penulisan dan struktur pseudocode untuk merancang algoritma tanpa harus menggunakan sintaks bahasa pemrograman yang spesifik. Mahasiswa juga akan memahami bagaimana instruksi didefinisikan dalam algoritma, dan bagaimana proses-proses ini bekerja secara berurutan untuk mencapai hasil akhir yang diinginkan.

B. Relevansi

Modul ini merupakan modul awal yang sangat penting sebelum mahasiswa mempelajari berbagai jenis algoritma dalam pemrograman yang lebih kompleks. Pada modul ini, mahasiswa akan mempelajari konsep dasar dan pengertian algoritma dengan penekanan pada penggunaan pseudocode sebagai alat untuk merancang dan menggambarkan algoritma secara sistematis. Pemahaman yang

diperoleh dalam modul ini akan memberikan landasan yang kuat bagi mahasiswa untuk menyusun solusi algoritmik yang terstruktur, yang nantinya akan diimplementasikan dalam bentuk diagram alur (flowchart) dan bahasa pemrograman pada modul-modul berikutnya.

Pengenalan dan penerapan pseudocode dalam modul ini memungkinkan mahasiswa untuk mengembangkan kemampuan berpikir logis dan sistematis dalam menyusun langkah-langkah pemecahan masalah. Mahasiswa akan mempelajari elemen-elemen dasar seperti tipe data, instruksi, aturan penulisan, dan proses dalam algoritma, yang menjadi dasar dalam perancangan solusi pemrograman. Dengan pemahaman ini, mahasiswa akan lebih siap untuk mendalami berbagai jenis algoritma dalam pemrograman yang lebih lanjut, baik dalam konteks algoritma pencarian, pengurutan, maupun algoritma yang lebih kompleks di modul-modul selanjutnya.

Dalam rangkaian modul ini, pemahaman tentang dasar algoritma dan penggunaan pseudocode sangat relevan terhadap pencapaian Sub-Capaian Pembelajaran Mata Kuliah (Sub-CPMK), karena memberikan dasar analitis bagi mahasiswa untuk dapat merancang algoritma secara logis dan efisien. Dengan pemahaman ini, akan membantu mahasiswa mengembangkan keterampilan dalam merancang algoritma yang akan diimplementasikan dalam berbagai bahasa pemrograman pada modul-modul selanjutnya.

C. Pembelajaran

Capaian pembelajaran yang diharapkan pada kegiatan belajar yang pertama ini adalah peserta kuliah akan mampu mengetahui dasar algoritma. Untuk mencapai kompetensi lulusan ini, peserta kuliah diharapkan membaca dan menyelesaikan Kegiatan Belajar 1 ini. Untuk mencapai capaian pembelajaran pada Kegiatan Belajar 1, peserta

kuliah harus mengikuti urutan kegiatan pembelajaran yang telah disiapkan dalam modul ini.

1. Uraian

1.1 Pendahuluan

Penulisan algoritma dapat dinyatakan dalam bahasa biasa, flowchart, pseudocode, atau berbagai bahasa pemrograman seperti C++ dan Java, dan berfokus pada tiga aspek utama: input, proses, dan output. Sederhananya, Konsep dasar algoritma mencakup pemahaman mengenai bagaimana suatu masalah dapat dipecah menjadi langkah-langkah kecil dan disusun secara logis agar dapat menghasilkan output yang diinginkan dari input yang diberikan. Penulisan algoritma dalam bentuk pseudocode, flowchart, ataupun bahasa yang biasa sebelum melakukan pemrograman, dapat sangat membantu dalam menganalisis masalah yang diberikan sehingga tahapan pemrograman dari awal sampai akhir dapat dipahami fungsinya agar menghasilkan output yang sesuai. Selain itu, kita juga dapat melihat keefektifan dan keefesienan suatu algoritma sebelum melakukan pemrograman.

1.2 Pseudocode

Pseudocode merupakan bahasa informal yang mirip dengan bahasa pemrograman untuk mendeskripsikan program. Bahasa ini biasa digunakan pada materi pembelajaran algoritma, sehingga pembaca tidak perlu mempelajari suatu bahasa pemrograman tertentu. Dalam penulisan pseudocode memang tidak ada aturan yang pasti akan tetapi penulisannya harus jelas dan logis. Berikut adalah beberapa kriteria penulisan yang baik dalam menuliskan pseudocode:

1. Tulis secara spesifik dan konsisten

Tulislah pseudocode kamu secara spesifik dan juga konsisten. Konsisten disini terletak pada penggunaan huruf kapital dan kecil. Kamu dapat menggunakan huruf kapital untuk kode perintah, misalnya IF, ELSE, dan THEN. Hal ini dapat membantu kamu saat nanti menulis kode program agar kamu tidak kebingungan untuk membedakan notasi dan komponen yang dinotasikan.

2. Gunakan indentasi

Walaupun penggunaan indentasi dalam pseudocode tidak diharuskan tetapi gunakanlah indentasi untuk memudahkan kamu dalam membaca notasi seperti if, for, dan while. Perlu kamu ketahui juga, indentasi ini sangat berpengaruh dalam beberapa bahasa pemrograman seperti bahasa pemrograman Python.

3. Buat dengan sederhana

Usahakan, buatlah kode semu kamu tetap sederhana hal itu untuk memudahkan kamu untuk menerjemahkannya menjadi kode program.

1.3 Contoh Penulisan Pseudocode

Pseudocode Program Mencari Nilai Maksimum dari Tiga Angka

Start

```
    Declare integer a, b, c, max  
    Print "Masukkan angka pertama: "  
    Input a  
    Print "Masukkan angka kedua: "  
    Input b  
    Print "Masukkan angka ketiga: "  
    Input c
```

```
max = a
If b > max Then
    max = b
End If
If c > max Then
    max = c
End If

Print "Angka terbesar adalah: ", max
End
```

2. Rangkuman

Algoritma dapat ditulis menggunakan bahasa sehari-hari, flowchart, pseudocode, atau berbagai bahasa pemrograman seperti C++ dan Java, dan umumnya berfokus pada tiga aspek utama: input, proses, dan output. Dengan algoritma, masalah dapat dipecah menjadi langkah-langkah kecil yang logis untuk mencapai output yang diinginkan dari input yang diberikan. Penulisan algoritma dalam bentuk pseudocode atau flowchart sebelum membuat kode program sangat membantu untuk menganalisis dan memahami tahapan pemrograman secara keseluruhan, serta untuk menilai efektivitas dan efisiensi algoritma. Pseudocode sendiri merupakan bahasa informal yang menyerupai bahasa pemrograman, bertujuan untuk memudahkan pemahaman tanpa harus menguasai bahasa pemrograman tertentu. Beberapa kriteria penulisan pseudocode yang baik meliputi konsistensi dalam penulisan, penggunaan indentasi untuk memudahkan pembacaan, dan menjaga

kesederhanaan notasi agar mudah diterjemahkan menjadi kode program.

3. Pustaka.

[1] Aji, Alham Fikri dan Gozali, William. Pemrograman Kompetitif Dasar. ISBN 978-602-6598-89-9.

[2] Setiawan, Rony. (2021). Kenali Pseudocode untuk Developer Pemula.

D. Penilaian Kegiatan Belajar

1. Penilaian

1.1 Latihan

1) Buatlah pseudocode dari sebuah program untuk menjumlahkan deret bilangan dari deret pertama hingga deret ke-n!

Format masukan :

Input	Output
5	15
10	55

2) Buatlah pseudocode dari sebuah program untuk menemukan nilai terbesar dari N angka yang diberikan!

Format masukan :

Input	Output
5	7
1 3 5 6 7	
3	-1

-1 -5 -6	
----------	--

1.2 Tantangan

- 1) Buatlah pseudocode untuk program yang menerima daftar nilai ujian siswa, menghitung nilai rata-rata, kemudian menentukan dan menampilkan predikat setiap nilai berdasarkan rentang berikut:

Nilai A jika nilai ≥ 85

Nilai B jika nilai ≥ 70 dan < 85

Nilai C jika nilai ≥ 50 dan < 70

Nilai D jika nilai < 50

Program juga harus menghitung persentase jumlah siswa yang mendapatkan nilai A, B, C, dan D.

Contoh input:

Daftar nilai = [90, 75, 60, 40, 85, 95, 70]

Output yang diharapkan:

1. Rata-rata nilai
Rata-rata = 73.57
2. Predikat untuk setiap nilai
Nilai: 90, Predikat: A
Nilai: 75, Predikat: B
Nilai: 60, Predikat: C

Nilai: 40, Predikat: D

Nilai: 85, Predikat: A

Nilai: 95, Predikat: A

Nilai: 70, Predikat: B

3. Persentase siswa untuk setiap predikat (A, B, C, D)

Persentase A = $(3 / 7) * 100 = 42.86\%$

Persentase B = $(2 / 7) * 100 = 28.57\%$

Persentase C = $(1 / 7) * 100 = 14.29\%$

Persentase D = $(1 / 7) * 100 = 14.29\%$

2. Umpam Balik dan Tindak Lanjut

- 1) Pseudocode dari sebuah program untuk menjumlahkan deret bilangan dari deret pertama hingga deret ke-n

Pseudocode program untuk menjumlahkan deret bilangan dari deret pertama hingga deret ke-n

Start

 Declare integer n, sum = 0

 Print "Masukkan angka n: "

 Input n

 For i = 1 to n Do

 sum = sum + i

 End For

 Print "Jumlah deret dari 1 hingga", n,
 "adalah:", sum

End

- 2) Pseudocode dari sebuah program untuk menemukan nilai terbesar dari N angka yang diberikan

Pseudocode program untuk menemukan nilai terbesar dari N angka yang diberikan

Start

 Declare integer n, max

 Input n

 Declare array bilangan[n]

 For i = 0 to n-1 Do

 Input bilangan[i]

 End For

 max = bilangan[0]

 For i = 1 to n-1 Do

 If bilangan[i] > max Then

 max = bilangan[i]

 End If

 End For

 Print max

End



ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA



MODUL III

Flowchart

Elly Warni
Ingrid Nurtanio
Amil Ahmad Ilham
Adnan
Anugrayani Bustamin

PRAKATA

Alhamdulillah, puji dan syukur kami panjatkan ke hadirat Allah SWT yang telah melimpahkan rahmat dan hidayah-Nya, sehingga modul ini dapat terselesaikan dengan baik. Shalawat serta salam senantiasa tercurah kepada junjungan kita, Nabi Muhammad SAW, beserta keluarga dan sahabat beliau yang telah membawa kita dari zaman kegelapan menuju terang benderang ilmu pengetahuan.

Modul ini disusun sebagai bagian dari rangkaian pembelajaran yang bertujuan untuk membantu mahasiswa memahami konsep dan penerapan algoritma dalam pemrograman, khususnya dalam membuat dan mengimplementasikan flowchart. Modul ini merupakan lanjutan dari modul sebelumnya yang membahas dasar-dasar algoritma, sehingga diharapkan mahasiswa memiliki pemahaman yang lebih mendalam dan terstruktur.

Tidak dapat dipungkiri bahwa proses penyusunan modul ini tidak lepas dari dukungan berbagai pihak. Oleh karena itu, kami mengucapkan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan masukan dan bimbingan dalam proses penyusunan modul ini. Terima kasih kami sampaikan kepada rekan-rekan dosen dan staf yang telah memberikan kritik dan saran yang membangun, sehingga modul ini dapat terselesaikan dengan baik.

Kami menyadari bahwa modul ini masih memiliki keterbatasan dan kekurangan. Oleh karena itu, kami sangat mengharapkan masukan, kritik, dan saran dari para pembaca untuk penyempurnaan di masa yang akan datang. Semoga modul ini dapat memberikan manfaat dan menjadi bahan referensi yang berharga bagi para mahasiswa dalam memahami konsep algoritma dan pemrograman.



Akhir kata, semoga upaya ini mendapat ridha Allah SWT dan memberikan kontribusi positif bagi kemajuan ilmu pengetahuan. Amin.

Makassar, 12 November 2024

Penulis

DAFTAR ISI

PRAKATA	i
DAFTAR ISI	iii
MODUL ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA.....	1
Pengantar	1
KEGIATAN BELAJAR 1	3
A. Deskripsi Singkat	3
B. Relevansi	4
C. Pembelajaran	5
D. Penilaian Kegiatan Belajar	14



UNIVERSITAS HASANUDDIN
FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK INFORMATIKA

Kode
Dokumen

RENCANA PEMBELAJARAN SEMESTER					
MATA KULIAH (MK)	KODE	Rumpun MK	BOBOT (sks)	SEMESTER	Tgl Penyusunan
Algoritma Pemrograman dan Struktur Data	23D12110604	Teknik Informatika	T=2 P=2	2	19-02-2023
OTORISASI	Pengembang RPS	Koordinator RMK	Ketua PRODI		
	Dr. Ir. Ingrid Nurtanio., MT Dr. Amil Ahmad Ilham., ST., M.IT Adnan., ST., MT., PhD Anugrayani Bustamin., ST., MT MT Elly Warni, S.T., M.T	Dr. Ir. Ingrid Nurtanio., MT	Prof. Dr. Indrabayu., ST., M.T., M.Bus.Sys		
Capaian Pembelajaran (CP)	CPL-PRODI yang dibebankan pada MK <i>Intended Learning Outcomes</i>				
	CPL 1	Memiliki dasar pengetahuan Teknik Informatika yang meliputi teori dan konsep dasar dari Ilmu Komputer, Matematika dan Statistika, Algoritma dan Pemrograman, Rekayasa Perangkat Lunak, Manajemen Informasi dan Ketahanan Digital, serta pengetahuan tingkat lanjut pada bidang-bidang khusus Teknik Informatika, seperti Kecerdasan Buatan, Data Science, Jaringan Komputer, Komputasi Awan dan Internet of Things.			
	CPL 3	Mampu mengaplikasikan pengetahuan bidang Teknik Informatika yang dipadankan dengan bidang ilmu lainnya untuk menganalisa dan mencari solusi dari berbagai masalah berbasis komputasi.			
	CPL 4	Mampu mendesain, mengimplementasikan dan mengevaluasi solusi berbasis komputasi dengan mengaplikasikan ilmu Teknik Informatika dan dasar-dasar pembangunan perangkat lunak.			
	Capaian Pembelajaran Mata Kuliah (CPMK)				
	CPMK	Setelah mengikuti Mata Kuliah Algoritma Pemrograman dan Struktur Data selama 1 (satu) semester, mahasiswa mampu menguasai dan menerapkan konsep teoritis bidang Informatika dan ilmu komputer secara dan mendalam serta memformulasikan penyelesaian masalah procedural dengan pemikiran logis dan sistematis secara mandiri dan terukur sebagai dasar pengembangan perangkat lunak.			

	CPL ⇒ Sub-CPMK	
	CPL 1	Sub-CPMK 1 : Mahasiswa mampu mendefinisikan konsep dan pengertian Algoritma
	CPL 1	Sub-CPMK 2 : Mahasiswa mampu mendeskripsikan Dasar Algoritma
	CPL 3	Sub-CPMK 3 : Mahasiswa mampu mengimplementasikan Flowchart
	CPL 1	Sub-CPMK 4 : Mahasiswa mampu memahami bentuk pemilihan dan perulangan
	CPL 4	Sub-CPMK 5 : Mahasiswa mampu mengimplementasikan prosedur, fungsi dan pemrosesan teks
	CPL 1	Sub-CPMK 6 : Mahasiswa mampu mempraktekkan konsep array
	CPL 4	Sub-CPMK 7 : Mahasiswa mampu menetapkan ciri dari teknik pencarian
	CPL 4	Sub-CPMK 8 : Mahasiswa mampu menetapkan ciri dari teknik sorting dan menganalisis complexitas algoritma
	CPL 1	Sub-CPMK 9 : Mahasiswa mampu memahami struct dan konsep linked list
	CPL 1	Sub-CPMK 10 : Mahasiswa mampu memahami Antrian dan Stack
	CPL 1	Sub-CPMK 11 : Mahasiswa mampu memahami teknik graph
	CPL 3	Sub-CPMK 12 : Mahasiswa mampu mengimplementasi konsep Tree
Deskripsi Singkat MK	<p>Mata kuliah Algoritma Pemrograman dan Struktur Data merupakan mata kuliah yang menyajikan konsep pemrograman secara mendalam terkait algoritma dan struktur data. Muatan mata kuliah merupakan dasar dari ilmu matematika, algoritma dan logika yang menjadi syarat utama pada mata kuliah Pemrograman lanjut . Mata kuliah ini disajikan untuk mahasiswa semester dua di Program Studi Teknik Informatika, Penanggungjawab mata kuliah adalah Laboratorium Komputer . Metode Pembelajaran bauran, teori dan praktik</p>	
Bahan Kajian / Materi Pembelajaran	<ol style="list-style-type: none"> 1. Konsep Algoritma 2. Dasar Algoritma (Aturan Penulisan Header, deklarasi dan deskripsi) 3. Flowchart (Simbol, variabel, algoritma cabang, runtunan) 4. Pemilihan (Alir Kontrol dan Pengulangan) 5. Prosedur, Fungsi dan Pemrosesan Teks 6. Array – Larik 7. Pencarian (Binary Search) 8. Pencarian (Sequential Search) 9. Bubble dan Insertion Sort 10. Merge dan selection sort 11. Quick sort dan complexita algoritma 12. Struct dan Linked List 13. Antrian dan Stack 	

	14. Graph berarah dan tidak berarah 15. Tree								
Pustaka	Utama :	1. Munir, Rinaldi dan Lidya, Leony. 2016. Algoritma dan Pemrograman dalam Bahasa Pascal, C dan C++ Edisi Keenam. Informatika : Bandung 2. Gozali, William dan Aji, Alham Fikri. 2014. Pemrograman Kompetitif Dasar. Ikatan Alumni Tim Olimpiade Komputer Indonesia. 3. Zakaria, Teddy M dan Prijono, Agus. 2005. Konsep dan Implementasi Struktur Data. Informatika : Bandung. 4. Niswar, M., Ilham, A., Zainuddin, Z., Adnan, A., P, A., Warni, E., Aswad, I., & Muslimin, Z. (2021). Sosialisasi Metode Berfikir Komputasional pada Pendidikan Dasar dan Menengah di Lingkup Sulawesi Selatan. URNAL EPAT eknologi erapan ntuk engabdian asyarakat, 4(1), 46-52. https://doi.org/10.25042/jurnal_tepat.v4i1.172							
	Pendukung:	Tuliskan pustaka pendukung jika ada, sebagai pengayaan literasi							
Dosen Pengampu	1. Dr. Ir. Ingrid Nurtanio., MT 2. Dr. Amil Ahmad Ilham., ST., M.IT 3. Adnan., ST., MT., PhD 4. Anugrayani Bustamin., ST., MT 5. Elly Warni, S.T., M.T								
Matakuliah syarat	Dasar Pemrograman Komputer								
				Bentuk Pembelajaran,					
Pekan Ke-	Sub-CPMK (Kemampuan akhir tiap tahapan belajar)	Penilaian	Indikator	Metode Pembelajaran, Penugasan Mahasiswa, [Estimasi Waktu]		Materi Pembelajaran [Pustaka]	Bobot Penilaian (%)		
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)		

1	Mahasiswa mampu mengetahui konsep dan pengertian Algoritma	Mengetahui konsep definisi, manfaat, mekanisme dan kontribusi Algoritma	Assignments	Lecture TM: 4 x 50 menit.	Lecture VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Konsep dan definisi algoritma 2. Mekanisme dan Kontribusi Algoritma	5%
2	Mahasiswa mampu memahami Dasar Algoritma CPL 3- SubCPMK 2	Memahami Dasar Algoritma (tipe data, aturan penulisan, proses, instruksi dan aksi)	quizzes	Lecture, case study TM: 4 x 50 menit.	Lecture, case study VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Dasar Algoritma (tipe data, aturan penulisan, proses, instruksi, dan aksi) Pustaka : [1]	5%
3	Mahasiswa mampu mengenali Flowchart CPL 4- SubCPMK 3	Mampu mengenali Flowchart (simbol, variabel, algoritma cabang, dan runtunan)	Assignments	Lecture, small group discussion TM: 4 x 50 menit.	Lecture, small group discussion VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit	Flowchart (Simbol, variabel, algoritma cabang, runtunan) Pustaka : [1]	7%

4	Mahasiswa mampu memahami bentuk pemilihan (alih kontrol dan pengulangan)	Mampu memahami bentuk pemilihan (alih kontrol dan pengulangan)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Pemilihan Kontrol Pengulangan) Pustaka : [1]	(Alir dan 7%
5	Mahasiswa mampu mengimplementasikan prosedur, fungsi dan pemrosesan teks	Mampu mengimplementasikan pemanggilan prosedur, pemanggilan fungsi dan pemrosesan teks	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Pemanggilan Prosedur 2. Pemanggilan Fungsi 3. Pemrosesan Teks Pustaka : [1]	7%
6	Mahasiswa mampu mempraktekkan konsep array	Mampu mempraktekkan konsep array seperti menginisialisasi array, mengisi elemen, mencari nilai maksimum dan minimum	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Menginisialisasi array 2. Mengisi elemen array 3. Mencari nilai maksimum dan minimum Pustaka : [1][2]	7%

					x.php?id_session=13610		
7	Mahasiswa mampu menetapkan ciri dari teknik pencarian	Mampu menetapkan ciri dari teknik pencarian ● Binary Search ● Sequential Search	Assignments quizzes	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Binary Search 2. Sequential Search Pustaka : [2][3]	7%
8	Evaluasi Tengah Semester / Ujian Tengah Semester						
9	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting ● Bubble Sort ● Insertion Sort	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Bubble Sort 2. Insertion Sort Pustaka : [2][3]	9%
10	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting Merge Sort Selection Sort	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit	1. Merge Sort Selection Sort Pustaka : [2][3]	9%

					https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610		
11	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting Quick Sort berserta complexitas Algoritmanya	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Quick Sort 2. Complexitas Algoritma Pustaka : [2][3]	9%
12	Mahasiswa mampu memahami struct dan konsep linked list	Mampu memahami struct dan konsep dan operasi dasar linked list (deklarasi simpul, penambahan simpul pada linked list awal, akhir dan tengah)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Konsep Struct dan implementasinya 2. Deklarasi list 3. Penambahan simpul pada list awal, akhir dan tengah Pustaka : [2][3]	7%
13	Mahasiswa mampu memahami Antrian dan Stack	Mampu memahami Antrian dan Stack dalam List	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit	Antrian dan Stack dalam list Pustaka : [2][3]	7%

					https://sikola. https://sikola.un has.ac.id/courses /104D4224/inde x.php?id_session =13610		
14	Mahasiswa mampu memahami teknik graph	mampu memahami teknik graph (graph berarah dan tidak berarah menggunakan linked list)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.un has.ac.id/courses /104D 4224/index.php? id_session=1361 0	1. Graph berarah Graph tidak berarah Pustaka : [2][3]	7%
15	Mahasiswa mampu mengimplementasi konsep Tree	Mampu mengimplementasi konsep Tree menggunakan linked list	Assignments quizzes	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.un has.ac.id/courses /104D4224/inde x.php?id_session =13610	Konsep tree menggunakan linked List Pustaka : [2][3]	7%
16	Evaluasi Akhir Semester / Ujian Akhir Semester						

MODUL ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA

MODUL III

FLOWCHART



Pengantar

Modul ini dirancang untuk membantu mahasiswa memahami dasar-dasar flowchart, termasuk simbol, variabel, algoritma cabang, dan runtunan yang digunakan dalam pemrograman. Dalam rangka mencapai capaian pembelajaran mata kuliah (CPMK) yang telah ditetapkan, mahasiswa akan diarahkan untuk mengenali dan mengidentifikasi elemen-elemen penting dalam pembuatan flowchart.

Pada Sub-CPMK kali ini, mahasiswa diharapkan mampu mengenali dan memahami fungsi dari simbol-simbol dalam flowchart serta bagaimana variabel dan algoritma digunakan dalam merancang proses yang runtut dan terstruktur. Kegiatan belajar akan dilakukan melalui diskusi kelompok kecil dan pembelajaran melalui pemberian materi serta didukung oleh platform pembelajaran daring.

Kegiatan pembelajaran ini akan mencakup,

- Tatap Muka (TM): 4 sesi, masing-masing 50 menit.
- Virtual Classroom (VC): 4 sesi, masing-masing 50 menit.
- Praktikum Terbimbing (PT): 4 sesi, masing-masing 60 menit.
- Bimbingan Mandiri (BM): 4 sesi, masing-masing 60 menit.

Sumber pembelajaran yang dapat diakses untuk mendukung pemahaman lebih lanjut tersedia di platform sikola, dengan pustaka referensi yang telah disediakan.

Setelah mengikuti modul ini, mahasiswa diharapkan dapat mencapai tingkat pemahaman yang cukup, setidaknya 80%, sebelum melanjutkan ke materi berikutnya.

KEGIATAN BELAJAR 1

FLOWCHART

A. Deskripsi Singkat

Mahasiswa yang mempelajari flowchart seringkali menghadapi tantangan dalam memahami elemen-elemen dasar seperti simbol, aturan, algoritma cabang, dan runtunan. Pemahaman ini penting karena flowchart merupakan salah satu alat visual yang digunakan dalam perancangan algoritma dan pemrograman untuk menggambarkan alur logika secara sistematis dan terstruktur. Pada kegiatan belajar pertama ini, peserta kuliah akan diperkenalkan dengan konsep dasar flowchart dan penerapannya dalam menyusun proses pemrograman.

Kegiatan belajar pertama ini akan mengantar peserta kuliah untuk memahami komponen dasar dalam pembuatan flowchart, seperti simbol-simbol yang digunakan untuk merepresentasikan operasi, keputusan, dan alur logika. Pada tahap ini, pembelajaran akan dikenalkan dengan simbol-simbol umum seperti persegi panjang untuk proses, belah ketupat untuk keputusan, dan panah untuk menunjukkan arah aliran.

Teori pertama yang akan dipaparkan dalam kegiatan belajar ini adalah pengenalan simbol dan variabel dalam flowchart. Simbol ini digunakan untuk memberikan representasi visual dari setiap langkah dalam algoritma yang dirancang. Sebagai contoh, simbol lingkaran menunjukkan titik awal atau akhir dari sebuah proses, dan simbol persegi panjang mewakili langkah pemrosesan. Dengan pemahaman yang tepat terhadap simbol-simbol ini, mahasiswa diharapkan dapat mengembangkan kemampuan dalam menyusun flowchart yang benar dan efektif.

Struktur dasar flowchart umumnya mengikuti alur logika yang sistematis, dimulai dari titik awal, diikuti oleh langkah-langkah

pemrosesan, hingga mencapai titik akhir. Dalam kegiatan ini, mahasiswa juga akan mempelajari cara merancang flowchart yang terdiri dari cabang keputusan dan alur yang bercabang, yang sering digunakan untuk situasi yang memerlukan pemilihan kondisi tertentu.

Dengan pemahaman yang tepat, flowchart dapat membantu mahasiswa memvisualisasikan alur logika sebelum diimplementasikan dalam kode pemrograman, sehingga meminimalisir kesalahan dan meningkatkan efisiensi dalam pemrograman. Dengan begitu, mahasiswa diharapkan mampu mengimplementasikan flowchart. Dengan deskripsi ini, mahasiswa diharapkan dapat memahami dan mengaplikasikan flowchart sebagai alat bantu yang penting dalam perancangan algoritma yang sistematis.

B. Relevansi

Materi dalam modul kedua sebelumnya merupakan dasar pemahaman terhadap algoritma yang menjadi fondasi dalam merancang logika pemrograman. Pada modul tersebut, mahasiswa mempelajari konsep dasar algoritma serta langkah-langkah logis yang membentuk suatu algoritma. Pemahaman ini tidak berdiri sendiri, melainkan didukung dengan penguasaan elemen-elemen dasar algoritma yang akan menjadi bekal dalam menyusun flowchart. Dengan landasan ini, mahasiswa dapat memahami bagaimana langkah-langkah logis dalam algoritma dapat divisualisasikan secara sistematis dalam bentuk flowchart.

Pemahaman ini sangat penting karena flowchart yang dipelajari dalam modul ini bertujuan untuk mengaplikasikan elemen-elemen algoritma ke dalam bentuk visual yang mudah dipahami. Flowchart memungkinkan mahasiswa untuk melihat alur logika dan proses yang diperlukan dalam menyelesaikan suatu masalah, sekaligus mengidentifikasi urutan yang logis dalam pemrograman. Dalam materi

ini, mahasiswa akan diperkenalkan dengan simbol-simbol dasar dalam flowchart, seperti simbol proses, keputusan, dan alur yang menghubungkan setiap langkah. Pengenalan ini akan memudahkan mahasiswa dalam menyusun diagram alur yang jelas dan terstruktur.

Teori dasar flowchart dalam modul ini akan menjadi landasan penting bagi pemahaman pada modul keempat. Pada modul tersebut, mahasiswa akan memperdalam pengetahuan mengenai bentuk pemilihan dan perulangan dalam algoritma, yang merupakan konsep penting dalam pemrograman. Flowchart yang telah dipelajari dalam modul ini akan membantu mahasiswa dalam menggambarkan logika percabangan dan perulangan secara visual, sehingga mereka dapat merencanakan berbagai kemungkinan kondisi dalam program sebelum implementasi kode.

Dalam rangkaian modul ini, pemahaman flowchart memiliki relevansi yang kuat terhadap pencapaian Sub-Capaian Pembelajaran Mata Kuliah (Sub-CPMK). Modul ini tidak hanya melatih kemampuan mahasiswa dalam membuat flowchart yang tepat, tetapi juga mempersiapkan mereka untuk menangani konsep logika pemrograman yang lebih kompleks di modul berikutnya. Mahasiswa diharapkan memiliki keterampilan analitis yang lebih baik dalam merancang algoritma yang efisien dan sistematis, sehingga dapat diimplementasikan dengan tepat dalam kode pemrograman.

C. Pembelajaran

Capaian pembelajaran yang diharapkan pada kegiatan belajar yang pertama ini adalah peserta kuliah akan mampu mengimplementasikan flowchart. Untuk mencapai kompetensi lulusan ini, peserta kuliah diharapkan membaca dan menyelesaikan Kegiatan Belajar 1 ini. Untuk mencapai capaian pembelajaran pada Kegiatan Belajar 1, peserta

kuliah harus mengikuti urutan kegiatan pembelajaran yang telah disiapkan dalam modul ini.

1. Uraian

1.1 Pendahuluan

Flowchart merupakan diagram alir yang disajikan secara sistematis dengan tampilan grafis yang menggambarkan suatu proses dan logika dari kegiatan penanganan informasi yang memuat urutan-urutan atau langkah-langkah prosedur pada suatu program yang digunakan dalam penyelesaian suatu masalah. Flowchart menjadi salah satu cara penyajian algoritma secara grafis yang sangat membantu programmer maupun non-programmer dalam membangun langkah-langkah penyelesaian suatu masalah. Flowchart menjadi penting untuk dipelajari karena kelebihan dan fungsi penting yang dimilikinya, antara lain:

- **Merencanakan proses**, flowchart membantu dalam merencanakan langkah-langkah yang diperlukan dalam menyelesaikan suatu tugas dengan lebih terstruktur.
- **Memudahkan komunikasi**, flowchart membantu programmer, analis, dan pengguna dalam memahami alur kerja suatu sistem secara lebih mudah dan efektif.
- **Memperbaiki kesalahan**, dengan melihat flowchart, programmer dapat mengidentifikasi dan memperbaiki kesalahan dalam algoritma dengan lebih cepat.

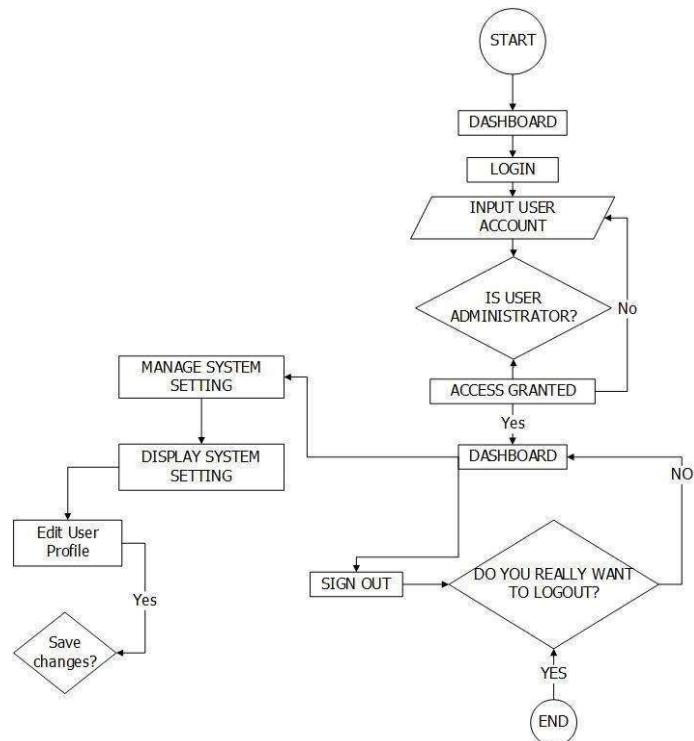
Berdasarkan jenisnya, flowchart dibagi menjadi 2, yaitu system flowchart dan program flowchart.

- 1) **System Flowchart**: Suatu flowchart yang menunjukkan arus pekerjaan dan menjelaskan urutan-urutan dari suatu prosedur yang ada di dalam sistem. Flowchart sistem

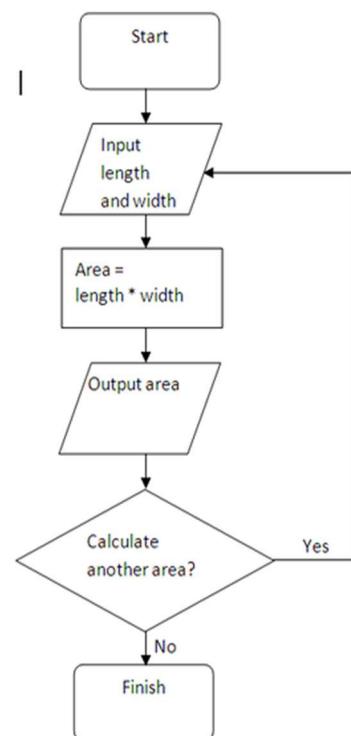
digunakan oleh analis sistem untuk menunjukkan berbagai proses, sub sistem, keluaran dan operasi pada data dalam suatu sistem.

- 2) **Program Flowchart:** Flowchart ini sering digunakan oleh programmer, karena mampu menunjukkan struktur program, aliran logika dan operasi yang dilakukan. Komponen ini merupakan bagian penting dari dokumentasi system.

System Flowchart



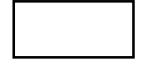
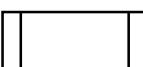
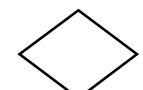
Program Flowchart



1.2 Simbol-simbol Flowchart

Flowchart merupakan representasi visual dari algoritma, dimana proses komputasi diwakili oleh simbol yang dikaitkan melalui anak panah yang merepresentasikan aliran algoritma. Simbol-simbol pada flowchart merujuk pada standarisasi yang sudah ditetapkan oleh American National Standards Institute

(ANSI). Simbol-simbol pada flowchart terbagi menjadi beberapa kategori dan sesuai fungsinya, yaitu input/output, pemrosesan, arah aliran/flowlines dan anotasi. Adapun simbol-simbol yang terdapat pada flowchart secara umum, yaitu sebagai berikut.

Simbol	Fungsi
	Terminator Permulaan/akhir program
	Flow Line Arah alir program
	Preparation Proses inisialisasi harga awal
	Process Proses perhitungan/pengolahan data
	Input/Output Data Proses input/output data, parameter, atau informasi
	Predefined Process (Sub Program) Permulaan sub program / proses menjalankan sub program
	Decision Penyeleksi data yang memberikan pilihan untuk langkah selanjutnya
	On Page Connector Penghubung bagian-bagian flowchart yang berada pada satu halaman
	Off Page Connector Penghubung bagian-bagian flowchart yang berada pada halaman berbeda

1.3 Aturan Pembuatan Flowchart

Sebenarnya tidak ada kaidah baku yang mengatur pembuatan suatu flowchart. Namun, ada beberapa petunjuk yang harus diperhatikan dalam pembuatan flowchart bagi seorang analis dan programmer, antara lain:

- Penggambaran Flowchart dilakukan dari halaman atas ke bawah dan dari kiri ke kanan.
- Awalan dan akhiran aktivitas harus ditentukan dengan jelas.
- Setiap aktivitas dan langkah harus diuraikan dengan menggunakan deskripsi kata kerja yang jelas.
- Langkah-langkah aktivitas harus berada pada urutan yang benar.
- Gunakan simbol-simbol flowchart yang standar.
- Aktivitas yang digambarkan dan didefinisikan harus penuh kehati-hatian dan pendefinisian setiap langkah atau alur yang digunakan harus bisa dipahami oleh pembacanya.

1.4 Teorema Terstruktur

Teorema terstruktur merupakan bagian dari konsep pemrograman terstruktur. Teorema ini berisi instruksi atau mekanisme yang terkait dengan kendali alur program. Teorema ini berperan penting dalam pengembangan algoritma dan pemrograman yang lebih mudah dipahami dan dikelola. Dengan teorema ini, program atau algoritma dapat dibangun dengan cara yang lebih sederhana dan terorganisir. Adapun manfaat teorema terstruktur, yaitu antara lain:

- **Readability:** Algoritma yang terstruktur lebih mudah dibaca dan dipahami oleh manusia.

- **Memudahkan debugging:** Karena alurnya jelas dan teratur, maka lebih mudah mendeteksi kesalahan atau bug.
- **Memfasilitasi Pemeliharaan Program:** Kode yang mengikuti prinsip terstruktur lebih mudah untuk diperbaiki dan diperbarui.

Terdapat tiga jenis aliran instruksi pada teorema terstruktur, yaitu:

1) Sequence

Sequence merupakan mekanisme aliran instruksi (instruction flows) dimana tiap instruksi dikerjakan secara berurutan sesuai dengan urutan penulisannya. Hal ini berarti tiap instruksi dikerjakan satu per satu sebanyak satu kali, urutan pelaksanaan instruksi sama dengan urutan penulisan program, dan instruksi terakhir merupakan akhir dari program.

Contoh:

Algoritma 1 Runtunan

{menghitung luas sebuah segitiga}

Deklarasi

real Alas,Tinggi,Luas;

Deskripsi

```
write("masukkan panjang alasnya:");
read(Alas);
write("masukkan tingginya:");
read(Tinggi);
Luas<-Alas*Tinggi/2;
write("Luas Segitiga=", Luas);
```

Penjelasan:

Urutan pengerjaan akan dimulai dari baris pertama sampai dengan baris terakhir. Jika terdapat satu baris yang diabaikan, maka akan menghasilkan error.

2) Selection

Selection merupakan mekanisme aliran instruksi (instructions flow) dimana suatu instruksi akan dikerjakan jika kondisi tertentu dipenuhi. Statement pertama dari selection akan dikerjakan jika kondisi bernilai benar, namun jika salah maka akan mengerjakan statement setelah keyword ‘else’.

Contoh:

Algoritma 2 Selection

{pemilihan kondisi menggunakan payung}

Deklarasi

 string Cuaca

 boolean Payung;

Deskripsi

 write(“masukkan kondisi cuaca:”);

 read(Cuaca);

 if Cuaca = “hujan” then

 Payung←True

 else

 Payung←False

Penjelasan:

Payung akan bernilai true jika cuaca memiliki value hujan, jika tidak maka payung akan bernilai false.

3) Repetition

Repetition merupakan mekanisme aliran instruksi (instructions flow) dimana suatu instruksi dikerjakan berulang-ulang sampai suatu kondisi yang ditetapkan tercapai.

Contoh:

Algoritma 3 Repetition
{mencetak angka 1 sampai 100}

Deklarasi
Int angka;
Deskripsi
angka<1
while (angka < 101) do
 write(angka);
 angka<=angka+1;

Penjelasan:

Pertama kali angka 1 akan dicetak, setelah itu nilai angka akan ditambah satu setiap repetisi, sehingga akan menampilkan angka 1 sampai 100.

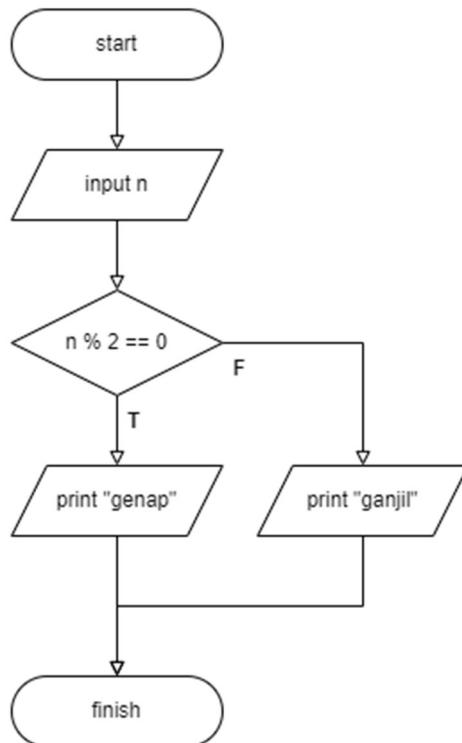
1.5 Contoh Flowchart

Buatlah flowchart yang:

- 1) Meminta pengguna untuk memasukkan sebuah bilangan bulat.

2) Memeriksa apakah bilangan tersebut adalah **genap** atau **ganjil**.

- Jika bilangan tersebut genap, tampilkan output: “**genap**”.
- Jika bilangan tersebut ganjil, tampilkan output: “**ganjil**”



2. Rangkuman

Flowchart adalah diagram alir yang menggambarkan proses atau logika suatu sistem secara visual dan sistematis, menggunakan simbol-simbol yang menunjukkan urutan langkah-langkah. Fungsi utamanya adalah membantu merencanakan proses, memudahkan komunikasi antara programmer dan pengguna, serta mempercepat identifikasi dan perbaikan kesalahan dalam algoritma. Terdapat dua jenis flowchart: system flowchart, yang menjelaskan alur pekerjaan dalam suatu sistem, dan program flowchart, yang memetakan struktur dan logika program. Simbol-simbol standard dalam

flowchart, seperti terminator, process, input/output, decision, dan connector, digunakan untuk menggambarkan alur dan keputusan dalam algoritma.

Teorema Terstruktur menyatakan bahwa semua algoritma dapat dibangun dari tiga struktur dasar: sequence (urutan), selection (pemilihan), dan repetition (pengulangan). Prinsip ini memastikan flowchart lebih mudah dipahami, diuji, dan dipelihara. Sebagai contoh, flowchart sederhana untuk menentukan apakah bilangan genap atau ganjil akan meminta input dari pengguna, mengecek kondisi bilangan, dan menampilkan hasilnya sesuai pemeriksaan.

3. Pustaka

[1] Munir, Rinaldi dan Lidya, Leony. 2016. Algoritma dan Pemrograman dalam Bahasa Pascal, C dan C++ Edisi Keenam. Informatika: Bandung.

[2] Putri, M. P., Barovih, G., Azdy, R. A., Yuniansyah, Saputra, A., Sriyeni, Y., Rini, A., & Admojo, F. T. (2022). Algoritma dan Struktur Data. Bandung: Widina Bhakti Persada Bandung. ISBN 978-623-459-182-8.

D. Penilaian Kegiatan Belajar

1. Penilaian

1.1 Latihan

- 1) Buat algoritma (penyajian dalam bentuk flowchart) beserta source programnya untuk “Menghitung nilai rata-rata dari 3 nilai bertipe integer dan output nilai rata-rata bertipe float”!
- 2) Buat algoritma (penyajian dalam bentuk flowchart) beserta source programnya untuk.

Program akan menerima sebuah bilangan bulat N.

- Jika N merupakan satuan, cetak satuan.
- Jika N merupakan puluhan, cetak puluhan.
- Jika N merupakan ratusan, cetak ratusan.
- Jika N merupakan ribuan, cetak ribuan.
- Jika N merupakan puluh ribuan, cetak puluhribuan."!

1.2 Tantangan

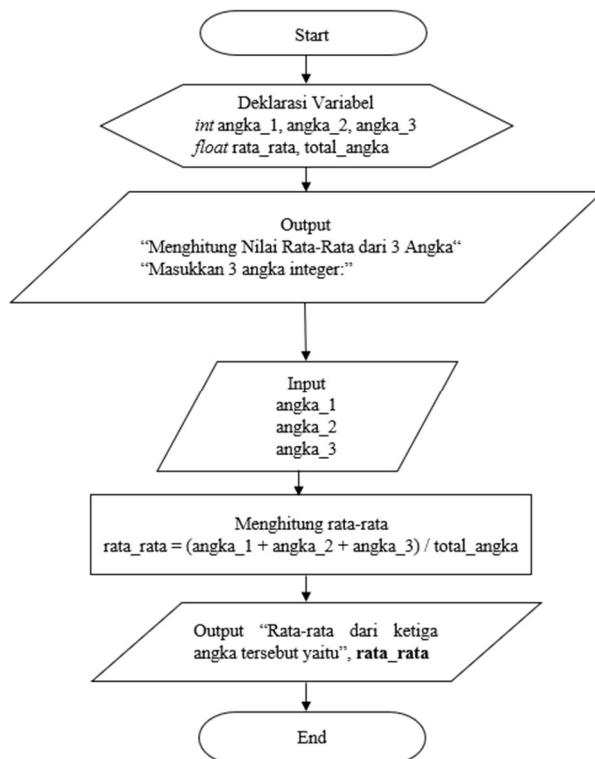
Buat algoritma (penyajian dalam bentuk flowchart dan pseudocode) beserta source programnya untuk.

- Program akan menerima N buah bilangan bulat: A_1 hingga A_N .
- Di antara bilangan-bilangan tersebut, tentukan bilangan terbesar dan bilangan terkecil.

2. Umpulan Balik dan Tindak Lanjut

1) Algoritma Nilai Rata-Rata

Flowchart



Source Code:

Source Code Nilai Rata Rata

```
#include <iostream>  
#include <iomanip>  
  
using namespace std;  
  
int main() {  
    int angka_1, angka_2, angka_3;  
    float rata_rata;
```

```
float total_angka = 3;

cout << "----- Menghitung Nilai Rata-Rata
dari 3 Angka ----- \n\n";

cout << "Masukkan angka pertama : "; cin >>
angka_1;

cout << "Masukkan angka kedua    : "; cin >>
angka_2;

cout << "Masukkan angka ketiga   : "; cin >>
angka_3;

rata_rata = (angka_1 + angka_2 + angka_3) /
total_angka;

cout << "Rata-rata dari ketiga angka
tersebut yaitu: "

<< fixed << setprecision(2) <<
rata_rata;

return 0;
}
```

2) Algoritma Cetak Bilangan

Source Code

Source Code Cetak Bilangan

```
#include <iostream>

using namespace std;

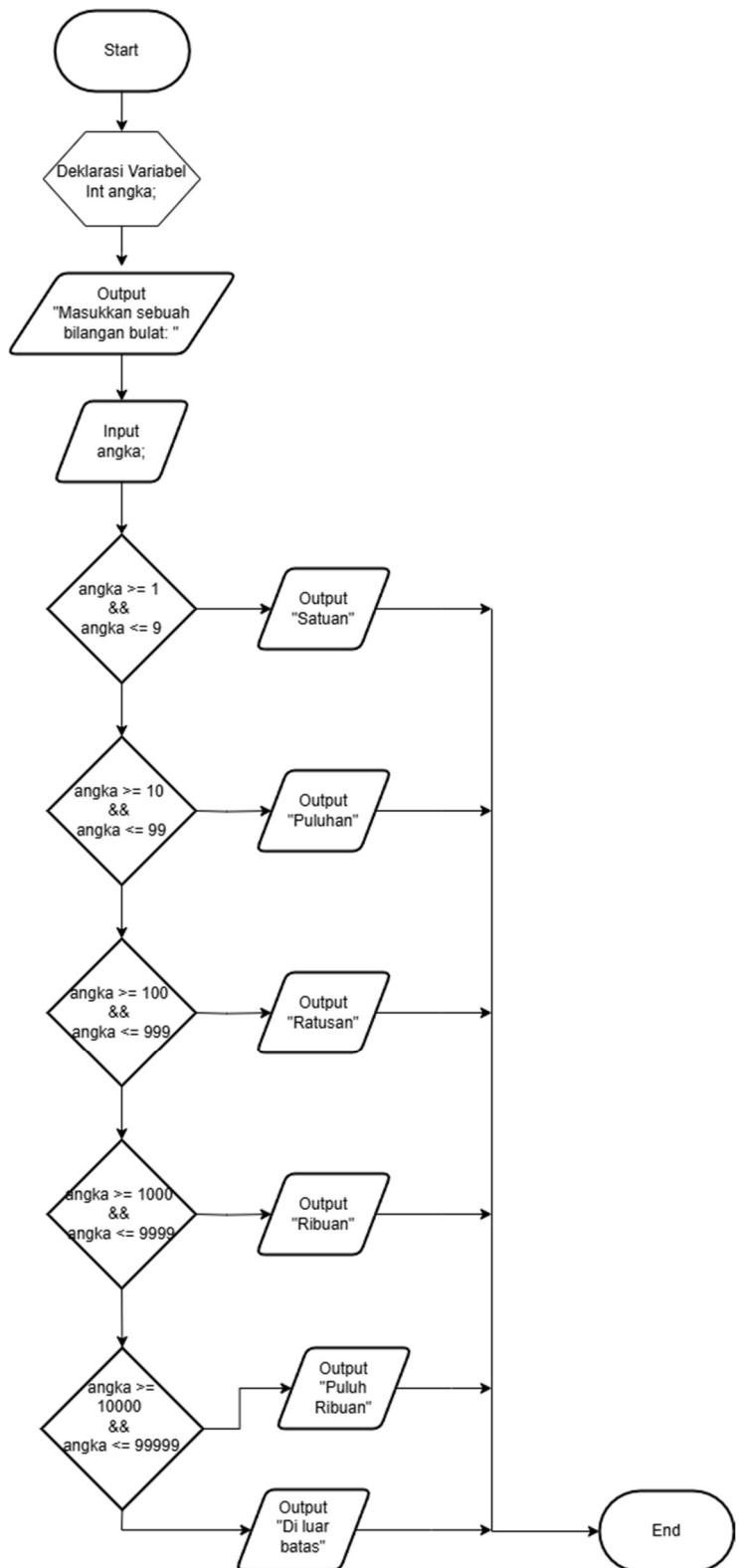
int main() {
    int N;
```

```
cout << "Masukkan sebuah bilangan bulat: ";
cin >> N;

// Memeriksa panjang bilangan dengan kondisi
if (N >= 1 && N <= 9) {
    cout << "Bilangan ini merupakan
satuan.\n";
} else if (N >= 10 && N <= 99) {
    cout << "Bilangan ini merupakan
puluhan.\n";
} else if (N >= 100 && N <= 999) {
    cout << "Bilangan ini merupakan
ratusan.\n";
} else if (N >= 1000 && N <= 9999) {
    cout << "Bilangan ini merupakan
ribuan.\n";
} else if (N >= 10000 && N <= 99999) {
    cout << "Bilangan ini merupakan puluh
ribuan.\n";
} else {
    cout << "Bilangan di luar jangkauan yang
ditentukan.\n";
}

return 0;
}
```

Flowchart





ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA



MODUL IV

Pemilihan Dan Perulangan

**Elly Warni
Ingrid Nurtanio
Amil Ahmad Ilham
Adnan
Anugrayani Bustamin**

PRAKATA

Alhamdulillah, puji dan syukur kami panjatkan ke hadirat Allah SWT yang telah melimpahkan rahmat dan hidayah-Nya, sehingga modul ini dapat diselesaikan dengan baik. Shalawat serta salam senantiasa tercurah kepada junjungan kita, Nabi Muhammad SAW, beserta keluarga dan sahabat beliau yang telah membawa kita dari zaman kegelapan menuju terang benderang ilmu pengetahuan.

Modul ini disusun sebagai bagian dari rangkaian pembelajaran yang bertujuan untuk membantu mahasiswa memahami konsep pemilihan dan perulangan dalam pemrograman. Kedua konsep ini sangat penting karena memungkinkan program untuk mengambil keputusan dan menjalankan proses berulang berdasarkan kondisi yang telah ditentukan. Dengan pemahaman yang baik mengenai pemilihan dan perulangan, mahasiswa diharapkan mampu mengembangkan program yang lebih dinamis dan efisien.

Tidak dapat dipungkiri bahwa proses penyusunan modul ini tidak lepas dari bantuan berbagai pihak. Oleh karena itu, kami mengucapkan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan masukan dan dukungan selama proses penyusunan modul ini. Ucapan terima kasih khusus kami sampaikan kepada rekan-rekan dosen dan staf yang telah memberikan kritik serta saran yang membangun, sehingga modul ini dapat tersusun dengan lebih baik.

Kami menyadari bahwa modul ini masih memiliki keterbatasan dan kekurangan. Oleh karena itu, kami sangat mengharapkan masukan, kritik, dan saran dari para pembaca untuk penyempurnaan di masa yang akan datang. Semoga modul ini dapat memberikan manfaat dan menjadi bahan referensi yang berharga bagi para mahasiswa dalam memahami konsep pemrograman.



Akhir kata, semoga upaya ini mendapat ridha Allah SWT dan memberikan kontribusi positif bagi kemajuan ilmu pengetahuan. Amin.

Makassar, 12 November 2024

Penulis

DAFTAR ISI

PRAKATA	i
DAFTAR ISI	iii
MODUL ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA.....	1
Pengantar.....	1
KEGIATAN BELAJAR 1.....	3
Pemilihan dan Perulangan.....	3
A. Deskripsi Singkat	3
B. Relevansi	4
C. Pembelajaran	5
D. Penilaian Kegiatan Belajar	18



UNIVERSITAS HASANUDDIN
FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK INFORMATIKA

Kode
Dokumen

RENCANA PEMBELAJARAN SEMESTER					
MATA KULIAH (MK)	KODE	Rumpun MK	BOBOT (sks)	SEMESTER	Tgl Penyusunan
Algoritma Pemrograman dan Struktur Data	23D12110604	Teknik Informatika	T=2	P=2	2 19-02-2023
OTORISASI	Pengembang RPS	Koordinator RMK	Ketua PRODI		Prof. Dr. Indrabayu., ST., M.T., M.Bus.Sys
	Dr. Ir. Ingrid Nurtanio., MT Dr. Amil Ahmad Ilham., ST., M.IT Adnan., ST., MT., PhD Anugrayani Bustamin., ST., MT Elly Warni, S.T., M.T	Dr. Ir. Ingrid Nurtanio., MT			
Capaian Pembelajaran (CP)	CPL-PRODI yang dibebankan pada MK <i>Intended Learning Outcomes</i>				
	CPL 1	Memiliki dasar pengetahuan Teknik Informatika yang meliputi teori dan konsep dasar dari Ilmu Komputer, Matematika dan Statistika, Algoritma dan Pemrograman, Rekayasa Perangkat Lunak, Manajemen Informasi dan Ketahanan Digital, serta pengetahuan tingkat lanjut pada bidang-bidang khusus Teknik Informatika, seperti Kecerdasan Buatan, Data Science, Jaringan Komputer, Komputasi Awan dan Internet of Things.			
	CPL 3	Mampu mengaplikasikan pengetahuan bidang Teknik Informatika yang dipadankan dengan bidang ilmu lainnya untuk menganalisa dan mencari solusi dari berbagai masalah berbasis komputasi.			
	CPL 4	Mampu mendesain, mengimplementasikan dan mengevaluasi solusi berbasis komputasi dengan mengaplikasikan ilmu Teknik Informatika dan dasar-dasar pembangunan perangkat lunak.			
	Capaian Pembelajaran Mata Kuliah (CPMK)				
	CPMK	Setelah mengikuti Mata Kuliah Algoritma Pemrograman dan Struktur Data selama 1 (satu) semester, mahasiswa mampu menguasai dan menerapkan konsep teoritis bidang Informatika dan ilmu komputer secara dan mendalam serta memformulasikan penyelesaian masalah procedural dengan pemikiran logis dan sistematis secara mandiri dan terukur sebagai dasar pengembangan perangkat lunak.			

	CPL ⇒ Sub-CPMK	
	CPL 1	Sub-CPMK 1 : Mahasiswa mampu mendefinisikan konsep dan pengertian Algoritma
	CPL 1	Sub-CPMK 2 : Mahasiswa mampu mendeskripsikan Dasar Algoritma
	CPL 3	Sub-CPMK 3 : Mahasiswa mampu mengimplementasikan Flowchart
	CPL 1	Sub-CPMK 4 : Mahasiswa mampu memahami bentuk pemilihan dan perulangan
	CPL 4	Sub-CPMK 5 : Mahasiswa mampu mengimplementasikan prosedur, fungsi dan pemrosesan teks
	CPL 1	Sub-CPMK 6 : Mahasiswa mampu mempraktekkan konsep array
	CPL 4	Sub-CPMK 7 : Mahasiswa mampu menetapkan ciri dari teknik pencarian
	CPL 4	Sub-CPMK 8 : Mahasiswa mampu menetapkan ciri dari teknik sorting dan menganalisis complexitas algoritma
	CPL 1	Sub-CPMK 9 : Mahasiswa mampu memahami struct dan konsep linked list
	CPL 1	Sub-CPMK 10 : Mahasiswa mampu memahami Antrian dan Stack
	CPL 1	Sub-CPMK 11 : Mahasiswa mampu memahami teknik graph
	CPL 3	Sub-CPMK 12 : Mahasiswa mampu mengimplementasi konsep Tree
Deskripsi Singkat MK	<p>Mata kuliah Algoritma Pemrograman dan Struktur Data merupakan mata kuliah yang menyajikan konsep pemrograman secara mendalam terkait algoritma dan struktur data. Muatan mata kuliah merupakan dasar dari ilmu matematika, algoritma dan logika yang menjadi syarat utama pada mata kuliah Pemrograman lanjut . Mata kuliah ini disajikan untuk mahasiswa semester dua di Program Studi Teknik Informatika, Penanggungjawab mata kuliah adalah Laboratorium Komputer . Metode Pembelajaran bauran, teori dan praktik</p>	
Bahan Kajian / Materi Pembelajaran	<ol style="list-style-type: none"> 1. Konsep Algoritma 2. Dasar Algoritma (Aturan Penulisan Header, deklarasi dan deskripsi) 3. Flowchart (Simbol, variabel, algoritma cabang, runtunan) 4. Pemilihan (Alir Kontrol dan Pengulangan) 5. Prosedur, Fungsi dan Pemrosesan Teks 6. Array – Larik 7. Pencarian (Binary Search) 8. Pencarian (Sequential Search) 9. Bubble dan Insertion Sort 10. Merge dan selection sort 11. Quick sort dan complexita algoritma 12. Struct dan Linked List 13. Antrian dan Stack 	

	14. Graph berarah dan tidak berarah 15. Tree								
Pustaka	Utama :	1. Munir, Rinaldi dan Lidya, Leony. 2016. Algoritma dan Pemrograman dalam Bahasa Pascal, C dan C++ Edisi Keenam. Informatika : Bandung 2. Gozali, William dan Aji, Alham Fikri. 2014. Pemrograman Kompetitif Dasar. Ikatan Alumni Tim Olimpiade Komputer Indonesia. 3. Zakaria, Teddy M dan Prijono, Agus. 2005. Konsep dan Implementasi Struktur Data. Informatika : Bandung. 4. Niswar, M., Ilham, A., Zainuddin, Z., Adnan, A., P, A., Warni, E., Aswad, I., & Muslimin, Z. (2021). Sosialisasi Metode Berfikir Komputasional pada Pendidikan Dasar dan Menengah di Lingkup Sulawesi Selatan. URNAL EPAT eknologi erapan ntuk engabdian asyarakat, 4(1), 46-52. https://doi.org/10.25042/jurnal_tepat.v4i1.172							
	Pendukung:	Tuliskan pustaka pendukung jika ada, sebagai pengayaan literasi							
Dosen Pengampu	1. Dr. Ir. Ingrid Nurtanio., MT 2. Dr. Amil Ahmad Ilham., ST., M.IT 3. Adnan., ST., MT., PhD 4. Anugrayani Bustamin., ST., MT 5. Elly Warni, S.T., M.T								
Matakuliah syarat	Dasar Pemrograman Komputer								
				Bentuk Pembelajaran,					
Pekan Ke-	Sub-CPMK (Kemampuan akhir tiap tahapan belajar)	Penilaian	Indikator	Metode Pembelajaran, Penugasan Mahasiswa, [Estimasi Waktu]		Materi Pembelajaran [Pustaka]	Bobot Penilaian (%)		
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)		

1	Mahasiswa mampu mengetahui konsep dan pengertian Algoritma	Mengetahui konsep definisi, manfaat, mekanisme dan kontribusi Algoritma	Assignments	Lecture TM: 4 x 50 menit.	Lecture VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Konsep dan definisi algoritma 2. Mekanisme dan Kontribusi Algoritma	5%
2	Mahasiswa mampu memahami Dasar Algoritma CPL 3- SubCPMK 2	Memahami Dasar Algoritma (tipe data, aturan penulisan, proses, instruksi dan aksi)	quizzes	Lecture, case study TM: 4 x 50 menit.	Lecture, case study VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Dasar Algoritma (tipe data, aturan penulisan, proses, instruksi, dan aksi) Pustaka : [1]	5%
3	Mahasiswa mampu mengenali Flowchart CPL 4- SubCPMK 3	Mampu mengenali Flowchart (simbol, variabel, algoritma cabang, dan runtunan)	Assignments	Lecture, small group discussion TM: 4 x 50 menit.	Lecture, small group discussion VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit	Flowchart (Simbol, variabel, algoritma cabang, runtunan) Pustaka : [1]	7%

4	Mahasiswa mampu memahami bentuk pemilihan (alih kontrol dan pengulangan)	Mampu memahami bentuk pemilihan (alih kontrol dan pengulangan)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Pemilihan Kontrol Pengulangan) Pustaka : [1]	(Alir dan 7%
5	Mahasiswa mampu mengimplementasikan prosedur, fungsi dan pemrosesan teks	Mampu mengimplementasikan pemanggilan prosedur, pemanggilan fungsi dan pemrosesan teks	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Pemanggilan Prosedur 2. Pemanggilan Fungsi 3. Pemrosesan Teks Pustaka : [1]	7%
6	Mahasiswa mampu mempraktekkan konsep array	Mampu mempraktekkan konsep array seperti menginisialisasi array, mengisi elemen, mencari nilai maksimum dan minimum	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Menginisialisasi array 2. Mengisi elemen array 3. Mencari nilai maksimum dan minimum Pustaka : [1][2]	7%

					x.php?id_session=13610		
7	Mahasiswa mampu menetapkan ciri dari teknik pencarian	Mampu menetapkan ciri dari teknik pencarian • Binary Search • Sequential Search	Assignments quizzes	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Binary Search 2. Sequential Search Pustaka : [2][3]	7%
8	Evaluasi Tengah Semester / Ujian Tengah Semester						
9	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting • Bubble Sort • Insertion Sort	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Bubble Sort 2. Insertion Sort Pustaka : [2][3]	9%
10	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting Merge Sort Selection Sort	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit	1. Merge Sort Selection Sort Pustaka : [2][3]	9%

					https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610		
11	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting Quick Sort berserta complexitas Algoritmanya	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Quick Sort 2. Complexitas Algoritma Pustaka : [2][3]	9%
12	Mahasiswa mampu memahami struct dan konsep linked list	Mampu memahami struct dan konsep dan operasi dasar linked list (deklarasi simpul, penambahan simpul pada linked list awal, akhir dan tengah)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Konsep Struct dan implementasinya 2. Deklarasi list 3. Penambahan simpul pada list awal, akhir dan tengah Pustaka : [2][3]	7%
13	Mahasiswa mampu memahami Antrian dan Stack	Mampu memahami Antrian dan Stack dalam List	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit	Antrian dan Stack dalam list Pustaka : [2][3]	7%

					https://sikola. https://sikola.un has.ac.id/courses /104D4224/inde x.php?id_session =13610		
14	Mahasiswa mampu memahami teknik graph	mampu memahami teknik graph (graph berarah dan tidak berarah menggunakan linked list)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.un has.ac.id/courses /104D 4224/index.php? id_session=1361 0	1. Graph berarah Graph tidak berarah Pustaka : [2][3]	7%
15	Mahasiswa mampu mengimplementasi konsep Tree	Mampu mengimplementasi konsep Tree menggunakan linked list	Assignments quizzes	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.un has.ac.id/courses /104D 4224/index.php? id_session=1361 0	Konsep tree menggunakan linked List Pustaka : [2][3]	7%
16	Evaluasi Akhir Semester / Ujian Akhir Semester						

MODUL ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA

MODUL IV

PEMILIHAN DAN PERULANGAN

=====

Pengantar

Modul ini bertujuan untuk membantu mahasiswa memahami konsep dasar dari bentuk pemilihan (decision-making) dan perulangan (looping) dalam pemrograman. Kedua konsep ini merupakan elemen fundamental dalam pemrograman yang memungkinkan kontrol alur program, sehingga program dapat mengeksekusi perintah-perintah tertentu berdasarkan kondisi tertentu (pemilihan) atau menjalankan serangkaian perintah secara berulang (perulangan).

Pada kegiatan belajar kali ini, mahasiswa akan mempelajari bagaimana struktur pemilihan seperti `if`, `else`, dan `switch` bekerja dalam menentukan jalur eksekusi program. Mahasiswa juga akan diperkenalkan pada konsep perulangan seperti `for`, `while`, dan `do-while` yang memungkinkan program untuk mengulang proses secara efektif berdasarkan kondisi yang telah ditetapkan.

Kegiatan ini mencakup perkuliahan tatap muka, diskusi kelompok, dan praktikum untuk memperdalam pemahaman mahasiswa mengenai penggunaan kontrol alur dalam pemrograman. Mahasiswa diharapkan dapat menguasai materi ini melalui pengajaran selama 4x50 menit untuk tatap muka, dan kegiatan tambahan di virtual classroom (VC), praktikum terbimbing (PT), serta bimbingan mandiri (BM) masing-masing selama 4x60 menit.

Melalui modul ini, mahasiswa diharapkan mampu menerapkan bentuk pemilihan dan perulangan dalam program yang mereka buat, sehingga dapat menyusun alur logika yang kompleks dengan kontrol penuh terhadap jalannya program.

KEGIATAN BELAJAR 1

Pemilihan dan Perulangan

A. Deskripsi Singkat

Mahasiswa yang mempelajari konsep pemilihan dan perulangan dalam pemrograman seringkali menghadapi tantangan dalam memahami alur logika yang bercabang dan bersiklus. Pemahaman ini penting karena pemilihan dan perulangan merupakan elemen dasar dalam pemrograman yang memungkinkan program untuk mengambil keputusan dan melakukan tugas secara berulang berdasarkan kondisi tertentu. Kedua konsep ini memberikan kontrol penuh terhadap jalannya program, sehingga program dapat berjalan secara dinamis sesuai dengan input atau kondisi yang ada.

Pada kegiatan belajar ini, peserta kuliah akan diperkenalkan dengan konsep dasar pemilihan dan perulangan serta penerapannya dalam menyusun logika pemrograman. Pemilihan mencakup struktur `if`, `else`, dan `switch` yang memungkinkan program memilih jalur eksekusi yang berbeda sesuai dengan kondisi yang diberikan. Perulangan melibatkan struktur `for`, `while`, dan `do-while` yang memungkinkan pengulangan serangkaian perintah berdasarkan kondisi yang terus diperiksa atau sejumlah iterasi tertentu.

Struktur pemilihan dan perulangan ini digunakan untuk mengatasi situasi di mana keputusan dan pengulangan proses diperlukan, seperti dalam pengulangan tindakan atau pemrosesan data. Dengan pemahaman yang tepat terhadap konsep-konsep ini, mahasiswa diharapkan dapat mengembangkan kemampuan dalam menyusun program yang fleksibel dan efisien.

B. Relevansi

Pada modul sebelumnya yang berfokus pada flowchart, mahasiswa telah mempelajari cara untuk menggambarkan alur logika dan langkah-langkah algoritma secara visual. Pemahaman tentang flowchart memberikan landasan bagi mahasiswa untuk memahami dan merancang logika pemrograman yang sistematis. Dengan menguasai simbol-simbol dan alur dalam flowchart, mahasiswa memperoleh kemampuan dalam menyusun diagram yang mencerminkan urutan logis suatu algoritma. Hal ini menjadi penting karena flowchart berperan sebagai dasar untuk mengembangkan logika yang lebih kompleks, termasuk pemilihan dan perulangan, yang akan mereka pelajari dalam modul ini.

Modul ini kemudian memperdalam pemahaman dengan mengajarkan mahasiswa bagaimana menggunakan struktur pemilihan (seperti `if`, `else`, dan `switch`) untuk memungkinkan program mengambil keputusan berdasarkan kondisi tertentu. Selain itu, mahasiswa juga akan belajar tentang perulangan (seperti `for`, `while`, dan `do-while`), yang memungkinkan program mengulangi proses tertentu secara efisien. Kedua konsep ini akan memungkinkan mahasiswa untuk mengembangkan alur logika yang dinamis, di mana program dapat menyesuaikan jalur eksekusi atau mengulangi langkah-langkah tertentu sesuai kebutuhan.

Pemahaman yang kuat terhadap materi ini akan menjadi landasan penting bagi modul selanjutnya, di mana mahasiswa akan mempelajari prosedur, fungsi, dan pemrosesan teks. Dalam modul berikutnya, mahasiswa akan membutuhkan kemampuan untuk mengatur alur logika yang lebih kompleks, termasuk penggunaan fungsi dan prosedur yang memanfaatkan pemilihan dan perulangan dalam pengaturan kode yang modular. Dengan menguasai struktur kontrol ini, mahasiswa akan lebih

mudah mengimplementasikan logika yang fleksibel dan terstruktur dalam kode pemrograman.

Secara keseluruhan, modul ini berperan penting dalam mengembangkan keterampilan pemrograman mahasiswa, khususnya dalam hal pengendalian alur logika. Melalui pemahaman yang mendalam tentang pemilihan dan perulangan, mahasiswa akan mampu merancang program yang lebih efisien dan adaptif, yang merupakan bagian dari pencapaian Sub-Capaian Pembelajaran Mata Kuliah (Sub-CPMK) untuk menghasilkan lulusan dengan keterampilan pemrograman yang handal dan siap pakai.

C. Pembelajaran

Capaian pembelajaran yang diharapkan pada kegiatan belajar yang pertama ini adalah peserta kuliah akan mampu memahami algoritma pemilihan dan perulangan. Untuk mencapai kompetensi lulusan ini, peserta kuliah diharapkan membaca dan menyelesaikan Kegiatan Belajar 1 ini. Untuk mencapai capaian pembelajaran pada Kegiatan Belajar 1, peserta kuliah harus mengikuti urutan kegiatan pembelajaran yang telah disiapkan dalam modul ini.

1. Uraian

1.1 Pendahuluan

Pemilihan atau percabangan adalah struktur program yang digunakan untuk melakukan proses pengujian terhadap satu, dua atau lebih dari dua kondisi. Pemilihan statement-statement atau perintah-perintah yang akan dijalankan didasarkan atas kondisi tertentu. Statement atau perintah tertentu akan dijalankan apabila memenuhi kondisi atau ketentuan yang telah didefinisikan sebelumnya. Operator relasi biasanya digunakan untuk menyatakan kondisi suatu statement pemilihan. Operator ini digunakan untuk membandingkan hubungan antara dua buah

operand yang akan menghasilkan value bertipe boolean (True atau False). Adapun operator relasi yang terdapat pada pemrograman, yaitu:

Operator	Operasi
=	Sama dengan
!=	Tidak sama dengan
>	Lebih besar dari
>=	Lebih besar atau sama dengan
<	Lebih kecil dari
<=	Lebih kecil atau sama dengan

1.2 Struktur Pemilihan

Terdapat 4 macam struktur pemilihan, yaitu sebagai berikut:

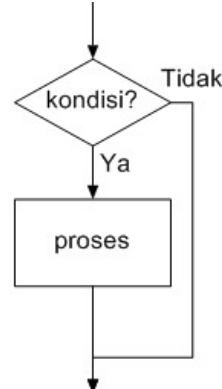
1) Statement if

Statement-if menentukan sebuah statement/proses yang akan dieksekusi jika dan hanya jika persyaratan boolean bernilai true. Pada statement if apabila kondisi tidak terpenuhi atau bernilai false maka program akan berhenti dan tidak menampilkan statement. Adapun bentuk umum dari statement if yaitu:

Pseudocode

```
IF <kondisi 1> THEN
    <statement>
ENDIF
```

Flowchart



Contoh:

Program 1. Bilangan Ganjil

```
int main() {  
    int angka;  
  
    cout << "Masukkan sebuah angka: ";  
    cin >> angka;  
  
    if (angka % 2 != 0) {  
        cout << angka << " adalah bilangan  
        ganjil.";  
    }  
  
    return 0;  
}
```

Pada program di atas, program akan meminta pengguna untuk memasukkan angka sembarang. Berdasarkan angka yang diinputkan tersebut, program akan melihat kondisi pemilihan, jika kondisi terpenuhi maka program akan menjalankan statement, tetapi apabila tidak terpenuhi, maka program akan selesai tanpa menjalankan statement apapun.

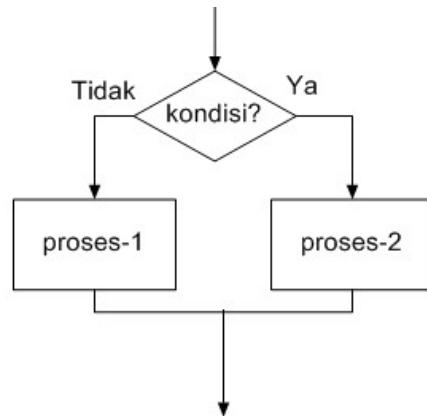
2) Statement if-else

Statement if-else digunakan apabila kita ingin mengeksekusi sebuah statement dengan kondisi true dan statement yang lain dengan kondisi false. Pada statement if-else apabila kondisi tidak terpenuhi atau bernilai false maka program akan mengeksekusi else statementnya. Adapun bentuk umum dari statement if-else yaitu:

Pseudocode

```
IF <kondisi 1> THEN
    <statement1>
ELSE
    <statement2>
ENDIF
```

Flowchart



Contoh:

Program 2. Bilangan Positif

```
int main() {
    int angka;

    cout << "Masukkan sebuah angka: ";
    cin >> angka;

    if (angka >= 0) {
        cout << angka << " adalah bilangan positif.";
    } else {
        cout << angka << " adalah bilangan negatif.";
    }

    return 0;
}
```

Pada program di atas, program akan meminta pengguna untuk memasukkan angka sembarang. Berdasarkan angka yang diinputkan tersebut, program akan melihat kondisi pemilihan, jika kondisi terpenuhi maka program akan menjalankan statement pertama, tetapi apabila tidak terpenuhi, maka program akan menuju else statement.

3) Statement if-else if

Statement pada bagian else dari blok if-else dapat menjadi struktur if else yang lain. Struktur seperti ini mengizinkan kita untuk membuat seleksi persyaratan yang lebih kompleks. Adapun bentuk umum dari statement if-else if yaitu:

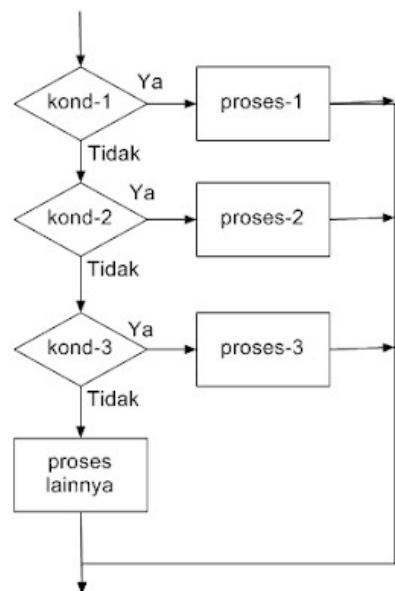
Pseudocode

```

IF <kondisi 1> THEN
    <statement1>
ELSE IF <kondisi 2> THEN
    <statement2>
ELSE IF <kondisi 3> THEN
    <statement3>
ELSE
    <statement4>
ENDIF

```

Flowchart



Statement if-else if memiliki banyak kondisi yang akan diuji kebenarannya. Statement ini terdiri dari beberapa blok program tergantung kondisi yang ada. Pada statement ini, statement else bersifat optional yang berarti dapat digunakan atau dapat juga dihilangkan berdasarkan kondisi atau kebutuhan program. Pada statement ini, jika kondisi 1 bernilai true, maka program akan mengeksekusi statement1 dan melewati statement yang lain. Jika kondisi 1 bernilai false, maka program akan mengecek kondisi else if pertama dan seterusnya hingga suatu kondisi bernilai true dan statement akan dijalankan. Apabila semua kondisi tidak terpenuhi, maka statement pada block else akan dieksekusi.

Contoh:

Program 3. Bining Nilai

```
int main() {  
    int nilai;  
  
    cout << "Masukkan nilai mahasiswa: ";  
    cin >> nilai;  
  
    if (nilai >= 85) {  
        cout << "Nilai adalah A.";  
    } else if (nilai >= 75) {  
        cout << "Nilai adalah B.";  
    } else if (nilai >= 65) {  
        cout << "Nilai adalah C";  
    } else {  
        cout << "Nilai adalah D";  
    }  
  
    return 0;  
}
```

Pada program di atas, program akan meminta pengguna untuk memasukkan nilai sembarang untuk seorang mahasiswa. Berdasarkan nilai yang diinputkan tersebut, program akan melihat beberapa kondisi menggunakan struktur if-else if.

- Jika nilai yang dimasukkan lebih besar atau sama dengan 80, maka kondisi pertama akan terpenuhi, dan program akan mencetak "Nilai adalah A".
- Jika kondisi pertama tidak terpenuhi, program akan memeriksa kondisi kedua, yaitu apakah nilai lebih besar atau sama dengan 65. Jika kondisi ini terpenuhi, program akan mencetak "Nilai adalah B".
- Jika kondisi pertama dan kedua tidak terpenuhi, program akan memeriksa kondisi ketiga, yaitu apakah nilai lebih besar atau sama dengan 50. Jika kondisi ini terpenuhi, program akan mencetak "Nilai adalah C".

- Apabila tidak ada kondisi yang terpenuhi (nilai kurang dari 50), program akan menuju else dan mencetak "Nilai adalah D".

Dengan demikian, program ini menggunakan struktur pemilihan untuk menentukan grade nilai berdasarkan rentang yang telah ditetapkan.

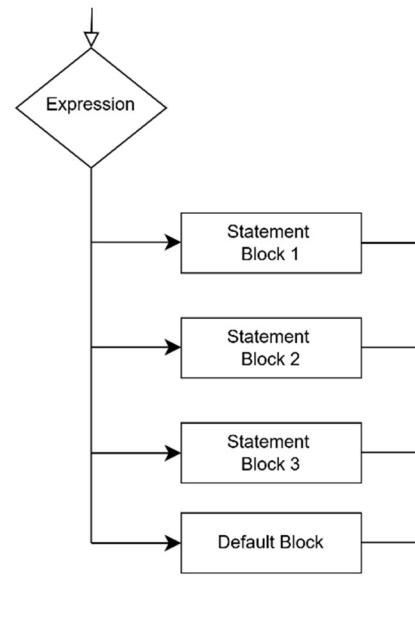
4) Statement switch

Statement Switch merupakan alternatif dari statement if-else if, biasanya digunakan pada masalah tertentu dimana aksi yang akan dilakukan hanya tergantung pada nilai dari satu macam variabel. Dengan kata lain, variabel yang menentukan ini mungkin memiliki banyak macam nilai dan setiap nilainya berkaitan dengan satu macam aksi.

Pseudocode

```
SWITCH <variabel>
    CASE <nilai 1>:
        <statement1>
        BREAK
    CASE <nilai 2>:
        <statement2>
        BREAK
    CASE <nilai 3>:
        <statement3>
        BREAK
    DEFAULT:
        <statement4>
ENDSWITCH
```

Flowchart



Contoh:

Program 4. Pemilihan (1 atau 2)

```
int main() {
    int pilihan;

    cout << "Masukkan angka (1 atau 2): ";
    cin >> pilihan;

    switch (pilihan) {
        case 1:
            cout << "Anda memilih angka satu.";
            break;
        case 2:
            cout << "Anda memilih angka dua.";
            break;
        default:
            cout << "Pilihan tidak valid." <<
    endl;
    }
}
```

Pada contoh di atas, program menggunakan struktur switch-case untuk mengelola pilihan berdasarkan angka yang dimasukkan oleh pengguna. Program ini dirancang untuk menerima input angka 1 atau 2 dari pengguna, dan kemudian menampilkan pesan yang sesuai dengan angka yang dipilih. Jika pengguna memasukkan angka selain 1 atau 2, program akan memberikan respons bahwa pilihan tidak valid.

1.3 Struktur Perulangan (Looping)

Struktur Perulangan adalah salah satu instruksi terpenting dalam pemrograman. Struktur ini digunakan untuk mengulang satu atau lebih pernyataan selama kondisi terpenuhi. Dengan struktur perulangan, programmer tidak perlu menulis kode program sebanyak jumlah pengulangan yang diperlukan. Pada masing-masing bahasa pemrograman mempunyai sintaks untuk struktur perulangan, tetapi secara garis besar struktur instruksi perulangan, yaitu:

- 1) Kondisi: suatu kondisi yang harus dipenuhi agar perulangan dapat terjadi.
- 2) Badan (body): deretan instruksi yang akan diulang-ulang pelaksanaannya.

Selain itu, perulangan juga mempunyai aturan yang harus dipenuhi, yaitu:

- 1) Inisialisasi adalah langkah persiapan untuk mengulang kondisi awal sebuah baris, seperti memasukkan nilai awal dalam variabel. Fase ini berjalan sebelum memasuki bagian perulangan.
- 2) Proses berjalan di dalam bagian perulangan yang berisi semua proses yang perlu dijalankan berulang kali.
- 3) Iterasi terjadi di dalam perulangan, ini merupakan sebuah kondisi tambahan untuk melanjutkan perulangan agar bisa terus berjalan
- 4) Terminasi adalah kondisi untuk berhenti dari sebuah perulangan, kondisi berhenti sangat penting untuk pengulangan, menghindari pengulangan tanpa batas. Kondisi perulangan adalah kondisi yang harus dipenuhi oleh algoritma yang sedang berjalan untuk memasuki blok perulangan.

Struktur perulangan memiliki sintaks yang berbeda-beda untuk setiap Bahasa pemrograman. Namun ada beberapa notasi struktur standar untuk perulangan ini yaitu for, while dan do-while. Berikut penjelasan masing-masing struktur standar perulangan.

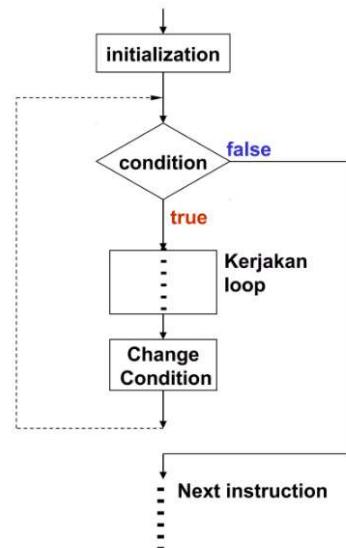
1) Perulangan For

Struktur For digunakan untuk mengulang perintah yang sudah ditentukan. Struktur ini memungkinkan programmer untuk menentukan seberapa sering perulangan akan berulang dimana jumlah perulangannya biasanya sudah ditentukan dari awal.

Pseudocode

```
FOR (inisialisasi; kondisi; iterasi)
    <statement>
ENDFOR
```

Flowchart



Contoh:

Program 5. For Loop (menampilkan angka 1 sampai 5)

```
int main() {
    for (int i = 1; i <= 5; i++) {
        cout << i << " ";
    }

    return 0;
}
```

Program ini menunjukkan penggunaan dasar for loop untuk menampilkan angka secara berurutan. Struktur for memungkinkan pengulangan tertentu dengan inisialisasi, kondisi, dan peningkatan variabel yang jelas. Blok kode akan terus dieksekusi selama kondisi memenuhi, yaitu i memiliki value kurang dari atau sama dengan 5.

2) Perulangan While

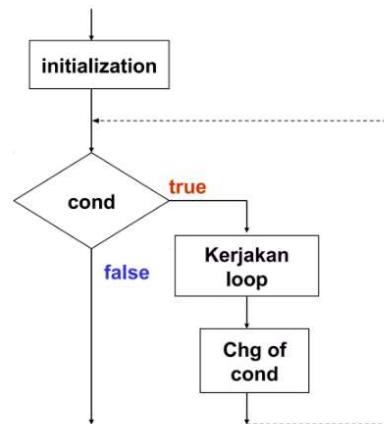
Struktur perulangan while digunakan untuk melakukan proses perulangan suatu statement terus menerus selama kondisi ungkapan logika pada while masih bernilai logika

benar. Pada struktur ini, kondisi akan dicek terlebih dahulu sebelum menjalankan perulangan atau biasa disebut entry-controlled loop.

Pseudocode

```
WHILE (kondisi) DO  
    <statement>  
ENDWHILE
```

Flowchart



Contoh:

Program 6. While Loop (menampilkan angka 1 sampai 5)

```
int main() {  
    int i = 1;  
  
    while (i <= 5) {  
        cout << i << " ";  
        i++;  
    }  
  
    return 0;  
}
```

Program ini menunjukkan penggunaan dasar while loop untuk menampilkan angka secara berurutan dari 1 hingga 5. While loop akan mengevaluasi kondisi terlebih dahulu sebelum menjalankan blok perintah, sehingga jika kondisi awal tidak terpenuhi, loop tidak akan berjalan sama sekali.

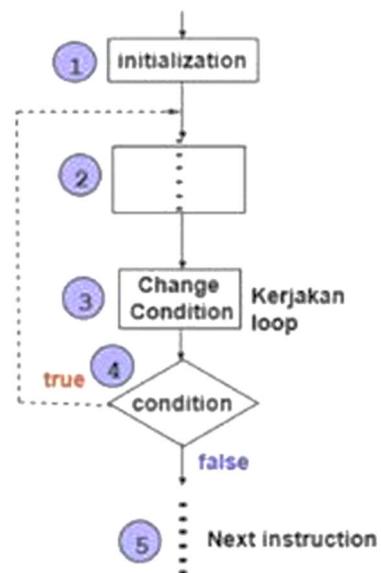
3) Perulangan Do While

Struktur perulangan do-while dieksekusi beberapa kali selama kondisi bernilai benar. Perbedaan antara perulangan while dan perulangan do-while adalah bahwa pernyataan pada perulangan do-while dieksekusi setidaknya satu kali karena kondisi do-while berada di bagian akhir program. Dengan kata lain, blok kode akan dieksekusi terlebih dahulu sebelum mengecek kondisi perulangan atau biasa disebut exit-controlled loop.

Pseudocode

```
DO  
    <statement>  
While <kondisi>  
END
```

Flowchart



Contoh:

Program 7. Do While Loop (menampilkan angka 1 sampai 5)

```
int main() {  
    int i = 1;  
    do {  
        cout << i << " ";
```

```
i++;
} while (i <= 5);

return 0;
}
```

Program ini menunjukkan penggunaan dasar **do-while loop** untuk menampilkan angka secara berurutan dari 1 hingga 5. Struktur do-while sangat berguna ketika kita ingin memastikan bahwa blok perintah dijalankan setidaknya satu kali, bahkan jika kondisi awal tidak terpenuhi.

2. Rangkuman

Struktur pemilihan/kondisi terbagi menjadi empat bagian, yaitu struktur if, Struktur if-else, Struktur if-else if, dan statement switch. Pada struktur kondisi yang menggunakan statement switch tipe data yang digunakan terbatas pada integer (byte, int, short) dan char. Algoritma Perulangan adalah algoritma yang mengulang atau mengulang langkah tertentu. Semua bahasa pemrograman memiliki sintaks struktur loop, tetapi secara umum memiliki tiga struktur loop, yaitu: for, while, dan do-while.

3. Pustaka

- [1] Munir, Rinaldi dan Lidya, Leony. 2016. Algoritma dan Pemrograman dalam Bahasa Pascal, C dan C++ Edisi Keenam. Informatika: Bandung.
- [2] Putri, M. P., Barovih, G., Azdy, R. A., Yuniansyah, Saputra, A., Sriyeni, Y., Rini, A., & Admojo, F. T. (2022). Algoritma dan Struktur Data. Bandung: Widina Bhakti Persada Bandung. ISBN 978-623-459-182-8.

D. Penilaian Kegiatan Belajar

1. Penilaian

1.1 Latihan

1) Buatlah algoritma flowchart beserta programnya yang meminta input berupa jumlah jam kerja dan golongan karyawan (A, B, C, atau D). Program akan menghitung dan menampilkan gaji karyawan berdasarkan aturan sebagai berikut:

- Upah per jam:
 - Golongan A: Rp15.000 per jam
 - Golongan B: Rp12.000 per jam
 - Golongan C: Rp10.000 per jam
 - Golongan D: Rp8.000 per jam
- Jika jam kerja melebihi 40 jam, karyawan berhak atas lembur dengan upah 1,5 kali upah per jam untuk setiap jam lembur.

Kriteria Perhitungan Gaji

- Jika jam kerja karyawan ≤ 40 jam:

$$\text{gaji} = \text{jam kerja} * \text{upah per jam}$$

- Jika jam kerja karyawan > 40 jam:

$$\text{gaji} = (40 * \text{upah per jam}) + (\text{jam lembur} * 1,5 * \text{upah per jam})$$

2) Buatlah program yang menampilkan pola segitiga bintang sebagai berikut:

pola 1

*

pola 2

1.2 Tantangan

- 1) Buatlah program iteratif yang menampilkan dan menghitung deret Fibonacci yang suku pertama dan keduanya ditentukan oleh pengguna. Deret Fibonacci adalah deret angka yang di mana setiap angka berikutnya merupakan penjumlahan dari dua angka sebelumnya. Contoh fibonacci dimana suku pertama yaitu 2 dan suku kedua yaitu 3:

2, 3, 5, 8, 13, 21, ...

Program harus meminta pengguna untuk memasukkan suku pertama, suku kedua, serta jumlah suku yang diinginkan dari deret Fibonacci. Program kemudian menampilkan deret Fibonacci hingga jumlah suku yang diminta.

2. Umpulan Balik dan Tindak Lanjut

- 1) Program Gaji Karyawan

Program Gaji Karyawan

```
#include <iostream>
using namespace std;

int main() {
    char golongan;
    int jam_kerja;
    float upah_per_jam;
    float gaji;

    //Meminta input golongan karyawan
```

```
cout << "Masukkan golongan karyawan (A, B,  
C, D): ";  
cin >> golongan;  
  
//Meminta input jumlah jam kerja  
cout << "Masukkan jumlah jam kerja: ";  
cin >> jam_kerja;  
  
//Menentukan upah per jam berdasarkan  
golongan  
switch (golongan) {  
    case 'A':  
        upah_per_jam = 15000;  
        break;  
    case 'B':  
        upah_per_jam = 12000;  
        break;  
    case 'C':  
        upah_per_jam = 10000;  
        break;  
    case 'D':  
        upah_per_jam = 8000;  
        break;  
    default:  
        cout << "Golongan tidak valid.\n";  
        return 1;  
}  
  
//Menghitung gaji  
if (jam_kerja <= 40) {  
    gaji = jam_kerja * upah_per_jam;  
} else {  
    int jam_lembur = jam_kerja - 40;  
    gaji = (40 * upah_per_jam) + (jam_lembur  
        * 1.5 * upah_per_jam);  
}  
  
// Menampilkan hasil gaji  
cout << "Gaji karyawan: Rp" << gaji << endl;  
  
return 0;  
}
```

2) Program Pola Bintang

a. Pola 1

Program Pola 1

```
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;

    for(int i = 1; i <= n; i++){
        for(int h = n; h > i; h--){
            cout << " ";
        }

        for(int j = 1; j <= 2*i-1; j++){
            cout << "*";
        }

        cout << endl;
    }
}
```

b. Pola 2

Program Pola 2

```
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;

    for(int i = n; i >= 1; i--){
        for(int h = n; h > i; h--){
            cout << " ";
        }

        for(int j = 1; j <= 2*i-1; j++){
            cout << "*";
        }
    }
}
```

```
    cout << endl;
}
}
```



ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA

MODUL V

*Prosedur, Fungsi,
Dan Pemrosesan
Teks*



Elly Warni
Ingrid Nurtanio
Amil Ahmad Ilham
Adnan
Anugrayani Bustamin

PRAKATA

Alhamdulillah, puji dan syukur kami panjatkan ke hadirat Allah SWT yang telah melimpahkan rahmat dan hidayah-Nya, sehingga modul ini dapat tersusun dengan baik. Shalawat dan salam semoga selalu tercurah kepada junjungan kita, Nabi Muhammad SAW, beserta keluarga dan para sahabat beliau, yang telah membimbing umat manusia menuju jalan ilmu pengetahuan dan kebenaran.

Modul ini disusun sebagai bahan pembelajaran untuk membantu mahasiswa memahami dan menguasai konsep prosedur, fungsi, dan pemrosesan teks dalam pemrograman. Ketiga konsep ini sangat penting karena membantu mahasiswa dalam membuat program yang modular, efisien, dan mudah dipelihara. Materi dalam modul ini juga akan menjadi landasan penting untuk memahami konsep array pada modul berikutnya.

Kami menyadari bahwa penyusunan modul ini tidak lepas dari bantuan berbagai pihak. Oleh karena itu, kami mengucapkan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan dukungan, masukan, dan saran dalam proses penyusunan modul ini. Ucapan terima kasih khusus kami sampaikan kepada rekan-rekan dosen dan staf yang telah membantu dengan kritik dan saran yang membangun.

Kami juga menyadari bahwa modul ini masih memiliki kekurangan. Oleh karena itu, kami sangat mengharapkan masukan dan kritik dari para pembaca untuk perbaikan di masa yang akan datang. Semoga modul ini dapat bermanfaat bagi mahasiswa dalam memahami konsep pemrograman, serta menjadi sarana untuk mencapai kompetensi yang diharapkan.

Akhir kata, semoga upaya ini mendapatkan ridha dari Allah SWT dan memberikan manfaat yang berkelanjutan bagi perkembangan ilmu pengetahuan.

Makassar, 12 November 2024

Penulis

DAFTAR ISI

PRAKATA	i
DAFTAR ISI	iii
MODUL ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA.....	1
Pengantar.....	1
KEGIATAN BELAJAR 1.....	3
Prosedur, Fungsi, dan Pemrosesan Teks.....	3
A. Deskripsi Singkat	3
B. Relevansi	3
C. Pembelajaran	4
D. Penilaian Kegiatan Belajar	18



UNIVERSITAS HASANUDDIN
FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK INFORMATIKA

Kode Dokumen

RENCANA PEMBELAJARAN SEMESTER					
MATA KULIAH (MK)	KODE	Rumpun MK	BOBOT (sks)	SEMESTER	Tgl Penyusunan
Algoritma Pemrograman dan Struktur Data	23D12110604	Teknik Informatika	T=2 P=2	2	19-02-2023
OTORISASI	Pengembang RPS	Koordinator RMK	Ketua PRODI		
	Dr. Ir. Ingrid Nurtanio., MT Dr. Amil Ahmad Ilham., ST., M.IT Adnan., ST., MT., PhD Anugrayani Bustamin., ST., MT MT Elly Warni, S.T., M.T	Dr. Ir. Ingrid Nurtanio., MT	Prof. Dr. Indrabayu., ST., M.T., M.Bus.Sys		
Capaian Pembelajaran (CP)	CPL-PRODI yang dibebankan pada MK <i>Intended Learning Outcomes</i>				
	CPL 1	Memiliki dasar pengetahuan Teknik Informatika yang meliputi teori dan konsep dasar dari Ilmu Komputer, Matematika dan Statistika, Algoritma dan Pemrograman, Rekayasa Perangkat Lunak, Manajemen Informasi dan Ketahanan Digital, serta pengetahuan tingkat lanjut pada bidang-bidang khusus Teknik Informatika, seperti Kecerdasan Buatan, Data Science, Jaringan Komputer, Komputasi Awan dan Internet of Things.			
	CPL 3	Mampu mengaplikasikan pengetahuan bidang Teknik Informatika yang dipadankan dengan bidang ilmu lainnya untuk menganalisa dan mencari solusi dari berbagai masalah berbasis komputasi.			
	CPL 4	Mampu mendesain, mengimplementasikan dan mengevaluasi solusi berbasis komputasi dengan mengaplikasikan ilmu Teknik Informatika dan dasar-dasar pembangunan perangkat lunak.			
	Capaian Pembelajaran Mata Kuliah (CPMK)				
	CPMK	Setelah mengikuti Mata Kuliah Algoritma Pemrograman dan Struktur Data selama 1 (satu) semester, mahasiswa mampu menguasai dan menerapkan konsep teoritis bidang Informatika dan ilmu komputer secara dan mendalam serta memformulasikan penyelesaian masalah procedural dengan pemikiran logis dan sistematis secara mandiri dan terukur sebagai dasar pengembangan perangkat lunak.			

	CPL ⇒ Sub-CPMK	
	CPL 1	Sub-CPMK 1 : Mahasiswa mampu mendefinisikan konsep dan pengertian Algoritma
	CPL 1	Sub-CPMK 2 : Mahasiswa mampu mendeskripsikan Dasar Algoritma
	CPL 3	Sub-CPMK 3 : Mahasiswa mampu mengimplementasikan Flowchart
	CPL 1	Sub-CPMK 4 : Mahasiswa mampu memahami bentuk pemilihan dan perulangan
	CPL 4	Sub-CPMK 5 : Mahasiswa mampu mengimplementasikan prosedur, fungsi dan pemrosesan teks
	CPL 1	Sub-CPMK 6 : Mahasiswa mampu mempraktekkan konsep array
	CPL 4	Sub-CPMK 7 : Mahasiswa mampu menetapkan ciri dari teknik pencarian
	CPL 4	Sub-CPMK 8 : Mahasiswa mampu menetapkan ciri dari teknik sorting dan menganalisis complexitas algoritma
	CPL 1	Sub-CPMK 9 : Mahasiswa mampu memahami struct dan konsep linked list
	CPL 1	Sub-CPMK 10 : Mahasiswa mampu memahami Antrian dan Stack
	CPL 1	Sub-CPMK 11 : Mahasiswa mampu memahami teknik graph
	CPL 3	Sub-CPMK 12 : Mahasiswa mampu mengimplementasi konsep Tree
Deskripsi Singkat MK	<p>Mata kuliah Algoritma Pemrograman dan Struktur Data merupakan mata kuliah yang menyajikan konsep pemrograman secara mendalam terkait algoritma dan struktur data. Muatan mata kuliah merupakan dasar dari ilmu matematika, algoritma dan logika yang menjadi syarat utama pada mata kuliah Pemrograman lanjut . Mata kuliah ini disajikan untuk mahasiswa semester dua di Program Studi Teknik Informatika, Penanggungjawab mata kuliah adalah Laboratorium Komputer . Metode Pembelajaran bauran, teori dan praktik</p>	
Bahan Kajian / Materi Pembelajaran	<ol style="list-style-type: none"> 1. Konsep Algoritma 2. Dasar Algoritma (Aturan Penulisan Header, deklarasi dan deskripsi) 3. Flowchart (Simbol, variabel, algoritma cabang, runtunan) 4. Pemilihan (Alir Kontrol dan Pengulangan) 5. Prosedur, Fungsi dan Pemrosesan Teks 6. Array – Larik 7. Pencarian (Binary Search) 8. Pencarian (Sequential Search) 9. Bubble dan Insertion Sort 10. Merge dan selection sort 11. Quick sort dan complexita algoritma 12. Struct dan Linked List 13. Antrian dan Stack 	

	14. Graph berarah dan tidak berarah 15. Tree								
Pustaka	Utama :	1. Munir, Rinaldi dan Lidya, Leony. 2016. Algoritma dan Pemrograman dalam Bahasa Pascal, C dan C++ Edisi Keenam. Informatika : Bandung 2. Gozali, William dan Aji, Alham Fikri. 2014. Pemrograman Kompetitif Dasar. Ikatan Alumni Tim Olimpiade Komputer Indonesia. 3. Zakaria, Teddy M dan Prijono, Agus. 2005. Konsep dan Implementasii Struktur Data. Informatika : Bandung. 4. Niswar, M., Ilham, A., Zainuddin, Z., Adnan, A., P, A., Warni, E., Aswad, I., & Muslimin, Z. (2021). Sosialisasi Metode Berfikir Komputasional pada Pendidikan Dasar dan Menengah di Lingkup Sulawesi Selatan. URNAL EPAT eknologi erapan ntuk engabdian asyarakat, 4(1), 46-52. https://doi.org/10.25042/jurnal_tepat.v4i1.172							
	Pendukung:	Tuliskan pustaka pendukung jika ada, sebagai pengayaan literasi							
Dosen Pengampu	1. Dr. Ir. Ingrid Nurtanio., MT 2. Dr. Amil Ahmad Ilham., ST., M.IT 3. Adnan., ST., MT., PhD 4. Anugrayani Bustamin., ST., MT 5. Elly Warni, S.T., M.T								
Matakuliah syarat	Dasar Pemrograman Komputer								
				Bentuk Pembelajaran,					
Pekan Ke-	Sub-CPMK (Kemampuan akhir tiap tahapan belajar)	Penilaian	Indikator	Metode Pembelajaran, Penugasan Mahasiswa, [Estimasi Waktu]		Materi Pembelajaran [Pustaka]	Bobot Penilaian (%)		
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)		

1	Mahasiswa mampu mengetahui konsep dan pengertian Algoritma	Mengetahui konsep definisi, manfaat, mekanisme dan kontribusi Algoritma	Assignments	Lecture TM: 4 x 50 menit.	Lecture VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Konsep dan definisi algoritma 2. Mekanisme dan Kontribusi Algoritma	5%
2	Mahasiswa mampu memahami Dasar Algoritma CPL 3- SubCPMK 2	Memahami Dasar Algoritma (tipe data, aturan penulisan, proses, instruksi dan aksi)	quizzes	Lecture, case study TM: 4 x 50 menit.	Lecture, case study VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Dasar Algoritma (tipe data, aturan penulisan, proses, instruksi, dan aksi) Pustaka : [1]	5%
3	Mahasiswa mampu mengenali Flowchart CPL 4- SubCPMK 3	Mampu mengenali Flowchart (simbol, variabel, algoritma cabang, dan runtunan)	Assignments	Lecture, small group discussion TM: 4 x 50 menit.	Lecture, small group discussion VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit	Flowchart (Simbol, variabel, algoritma cabang, runtunan) Pustaka : [1]	7%

4	Mahasiswa mampu memahami bentuk pemilihan (alih kontrol dan pengulangan)	Mampu memahami bentuk pemilihan (alih kontrol dan pengulangan)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Pemilihan Kontrol Pengulangan) Pustaka : [1]	(Alir dan 7%
5	Mahasiswa mampu mengimplementasikan prosedur, fungsi dan pemrosesan teks	Mampu mengimplementasikan pemanggilan prosedur, pemanggilan fungsi dan pemrosesan teks	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Pemanggilan Prosedur 2. Pemanggilan Fungsi 3. Pemrosesan Teks Pustaka : [1]	7%
6	Mahasiswa mampu mempraktekkan konsep array	Mampu mempraktekkan konsep array seperti menginisialisasi array, mengisi elemen, mencari nilai maksimum dan minimum	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Menginisialisasi array 2. Mengisi elemen array 3. Mencari nilai maksimum dan minimum Pustaka : [1][2]	7%

					x.php?id_session=13610		
7	Mahasiswa mampu menetapkan ciri dari teknik pencarian	Mampu menetapkan ciri dari teknik pencarian • Binary Search • Sequential Search	Assignments quizzes	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Binary Search 2. Sequential Search Pustaka : [2][3]	7%
8	Evaluasi Tengah Semester / Ujian Tengah Semester						
9	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting • Bubble Sort • Insertion Sort	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Bubble Sort 2. Insertion Sort Pustaka : [2][3]	9%
10	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting Merge Sort Selection Sort	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit	1. Merge Sort Selection Sort Pustaka : [2][3]	9%

					https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610		
11	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting Quick Sort berserta complexitas Algoritmanya	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Quick Sort 2. Complexitas Algoritma Pustaka : [2][3]	9%
12	Mahasiswa mampu memahami struct dan konsep linked list	Mampu memahami struct dan konsep dan operasi dasar linked list (deklarasi simpul, penambahan simpul pada linked list awal, akhir dan tengah)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Konsep Struct dan implementasinya 2. Deklarasi list 3. Penambahan simpul pada list awal, akhir dan tengah Pustaka : [2][3]	7%
13	Mahasiswa mampu memahami Antrian dan Stack	Mampu memahami Antrian dan Stack dalam List	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit	Antrian dan Stack dalam list Pustaka : [2][3]	7%

					https://sikola. https://sikola.un has.ac.id/courses /104D4224/inde x.php?id_session =13610		
14	Mahasiswa mampu memahami teknik graph	mampu memahami teknik graph (graph berarah dan tidak berarah menggunakan linked list)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.un has.ac.id/courses /104D 4224/index.php? id_session=1361 0	1. Graph berarah Graph tidak berarah Pustaka : [2][3]	7%
15	Mahasiswa mampu mengimplementasi konsep Tree	Mampu mengimplementasi konsep Tree menggunakan linked list	Assignments quizzes	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.un has.ac.id/courses /104D 4224/index.php? id_session=1361 0	Konsep tree menggunakan linked List Pustaka : [2][3]	7%
16	Evaluasi Akhir Semester / Ujian Akhir Semester						



MODUL ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA

Modul V

PROSEDUR, FUNGSI, DAN PEMROSESAN TEKS

Pengantar

Modul ini bertujuan untuk membantu mahasiswa memahami dan mengimplementasikan konsep dasar dari prosedur, fungsi, dan pemrosesan teks dalam pemrograman. Ketiga konsep ini merupakan elemen penting dalam pemrograman yang memungkinkan kode menjadi lebih modular, mudah dibaca, dan dapat digunakan kembali. Dengan memahami cara kerja prosedur dan fungsi, mahasiswa akan dapat membagi program menjadi bagian-bagian yang lebih kecil dan lebih terstruktur, sementara pemrosesan teks memungkinkan manipulasi data string secara lebih efektif.

Pada kegiatan belajar kali ini, mahasiswa akan mempelajari bagaimana cara mendefinisikan dan memanggil prosedur serta fungsi dalam program, termasuk parameter dan nilai kembali (return value). Mahasiswa juga akan diperkenalkan pada teknik dasar pemrosesan teks, seperti penggabungan (concatenation), pemotongan (substring), serta pemrosesan teks lainnya. Pemahaman ini sangat penting karena prosedur dan fungsi membantu menciptakan program yang lebih modular dan efisien, sedangkan pemrosesan teks berguna dalam banyak aplikasi yang memerlukan manipulasi data berbasis teks.

Kegiatan ini mencakup perkuliahan tatap muka, diskusi kelompok, dan praktikum untuk memperdalam pemahaman mahasiswa mengenai penggunaan prosedur, fungsi, dan pemrosesan teks dalam





pemrograman. Mahasiswa diharapkan dapat menguasai materi ini melalui pengajaran selama 4x50 menit untuk tatap muka, dan kegiatan tambahan di virtual classroom (VC), praktikum terbimbing (PT), serta bimbingan mandiri (BM) masing-masing selama 4x60 menit.

Melalui modul ini, mahasiswa diharapkan mampu mengimplementasikan prosedur dan fungsi dalam program yang mereka buat, serta melakukan berbagai manipulasi teks untuk memenuhi kebutuhan pemrograman. Mahasiswa juga akan memahami bagaimana penggunaan prosedur dan fungsi dapat meningkatkan efisiensi serta keterbacaan kode yang mereka tulis.

KEGIATAN BELAJAR 1

Prosedur, Fungsi, dan Pemrosesan Teks

A. Deskripsi Singkat

Materi ini bertujuan untuk membantu mahasiswa menguasai konsep prosedur, fungsi, dan pemrosesan teks dalam pemrograman, yang merupakan lanjutan dari materi pemilihan dan perulangan pada modul sebelumnya. Pada modul ini, mahasiswa akan belajar cara mendefinisikan dan memanggil prosedur serta fungsi untuk membuat kode yang lebih modular, terstruktur, dan mudah dipahami. Melalui prosedur dan fungsi, mahasiswa dapat mengorganisir kode ke dalam blok-blok yang dapat digunakan kembali dan mempermudah pemeliharaan program.

Selain itu, mahasiswa akan mempelajari teknik dasar pemrosesan teks yang memungkinkan manipulasi string, seperti penggabungan, pemotongan, dan pemrosesan string lainnya. Keterampilan ini sangat penting dalam aplikasi yang membutuhkan pengolahan data berbasis teks, seperti aplikasi pengolahan dokumen, analisis data teks, atau pengembangan antarmuka pengguna.

Pemahaman tentang prosedur, fungsi, dan pemrosesan teks akan menjadi fondasi yang kuat bagi mahasiswa saat mempelajari konsep array pada modul berikutnya, karena modularitas dan kemampuan manipulasi data sangat diperlukan dalam mengelola struktur data yang lebih kompleks.

B. Relevansi

Pada modul sebelumnya, mahasiswa mempelajari tentang pemilihan dan perulangan, yang memberikan dasar bagi mereka untuk memahami

alur logika dalam program. Kemampuan untuk mengatur alur dengan kondisi dan pengulangan menjadi landasan yang penting sebelum mahasiswa melangkah ke konsep prosedur dan fungsi, yang akan membantu mereka membuat program yang lebih modular dan mudah dipelihara. Melalui prosedur dan fungsi, mahasiswa dapat membagi program menjadi bagian-bagian yang lebih kecil dan spesifik, sehingga meningkatkan keterbacaan dan efisiensi kode.

Selain itu, materi tentang pemrosesan teks memungkinkan mahasiswa untuk memahami dan mengaplikasikan teknik manipulasi string dalam program. Kemampuan ini sangat berguna dalam banyak aplikasi pemrograman, seperti pengolahan data pengguna, pemrosesan dokumen, atau pengembangan antarmuka yang interaktif.

Pemahaman yang kuat terhadap prosedur, fungsi, dan pemrosesan teks akan mendukung pencapaian Sub-CPMK berikutnya, di mana mahasiswa akan belajar tentang array. Konsep array memungkinkan penyimpanan dan pengelolaan data dalam skala besar secara terstruktur, dan keterampilan modularisasi dengan prosedur dan fungsi akan memudahkan mahasiswa dalam mengelola data yang lebih kompleks di modul tersebut.

Secara keseluruhan, setelah mempelajari modul ini, mahasiswa diharapkan dapat mengimplementasikan prosedur, fungsi, dan pemrosesan teks dalam program yang mereka buat. Keterampilan ini akan membantu mereka mencapai kompetensi dalam merancang program yang lebih efisien, terstruktur, dan mampu menangani data teks secara efektif, serta mempersiapkan mereka untuk memahami konsep-konsep lanjutan pada modul berikutnya.

C. Pembelajaran

Capaian pembelajaran yang diharapkan pada kegiatan belajar yang pertama ini adalah peserta kuliah akan mampu memahami dan

mengimplementasikan prosedur, fungsi, dan pemrosesan teks dalam pemrograman. Untuk mencapai kompetensi lulusan ini, peserta kuliah diharapkan membaca dan menyelesaikan Kegiatan Belajar 1 ini. Untuk mencapai capaian pembelajaran pada Kegiatan Belajar 1, peserta kuliah harus mengikuti urutan kegiatan pembelajaran yang telah disiapkan dalam modul ini.

1. Uraian

1.1 Pendahuluan

Procedure dan function merupakan instruksi yang sering dijumpai dalam berbagai aktivitas, tidak hanya dalam kehidupan sehari hari tetapi juga dalam pemrograman. Secara umum procedure digunakan untuk fungsi yang tidak mengembalikan nilai, sedangkan function biasanya ditandai untuk fungsi yang mengembalikan nilai. Prosedur dan fungsi ada dalam pemrograman untuk membantu mengorganisir kode agar lebih terstruktur, modular, dan mudah dikelola. Penggunaan prosedur dan fungsi memiliki beberapa manfaat, antara lain:

- **Modularisasi:** Suatu program yang besar dan kompleks dapat dibagi ke dalam beberapa prosedur atau fungsi sehingga menjadi bagian yang mudah dikerjakan.
- **Simplifikasi:** Dalam suatu program, sering diperlukan suatu tugas yang harus dikerjakan berulang-ulang dengan nilai-nilai variabel yang berbeda. Agar tidak merepotkan maka tugas ini cukup ditulis sekali saja dalam bentuk prosedur atau fungsi yang kemudian dipanggil berulang-ulang sesuai kebutuhan.

- **Readability:** Dengan adanya prosedur dan fungsi, kode menjadi lebih mudah dipahami karena setiap prosedur atau fungsi memiliki tujuan spesifik.

1.2 Deklarasi dan Pemanggilan Prosedur

Prosedur merupakan istilah lain dari fungsi yang tidak mengembalikan nilai dan biasanya ditandai dengan keyword void. Prosedur memiliki struktur yang terdiri dari tiga bagian yaitu bagian judul, bagian deklarasi dan bagian algoritma (badan prosedur).

Sintaks:

```
Void NamaProsedur (DaftarParameter)
{
    /*Code atau Badan Prosedur*/
}
```

Penjelasan:

- **Void**, Kata kunci void digunakan sebagai tipe data pengembalian (return type) dari prosedur. Karena prosedur tidak mengembalikan nilai (nilai kembalian kosong), maka menggunakan void sebagai tipe datanya.
- **NamaProsedur**, Ini adalah nama dari prosedur, yang berfungsi sebagai identitas prosedur. Nama ini digunakan untuk memanggil prosedur di bagian lain dari program. Nama prosedur sebaiknya singkat tetapi mendeskripsikan fungsinya, seperti tampilkanPesan, cetakLuas, dll.
- **DaftarParameter**, Parameter adalah variabel yang digunakan untuk menerima nilai yang dikirimkan ketika

prosedur dipanggil. Parameter ditempatkan dalam tanda kurung () setelah nama prosedur.

- **Badan Prosedur { ... }**, Blok { ... } adalah tubuh atau badan dari prosedur, yang berisi kode yang akan dijalankan setiap kali prosedur ini dipanggil. Di dalam badan prosedur ini, kita dapat menuliskan instruksi atau logika yang diinginkan.
- Karena tipe data prosedur adalah void, tidak ada perintah return yang mengembalikan nilai dari prosedur ini.

Pemanggilan prosedur dapat digunakan untuk pertukaran data atau informasi. Program yang dipanggil dapat memberikan masukan (input) atau luaran (output), contohnya welcome("Anto").

Contoh:

Program 1. Prosedur Sambutan

```
#include <iostream>
using namespace std;

void tampilanSambutan(string nama) {
    cout << "Selamat datang, " << nama << "!" <<
endl;
}

int main() {
    string nama;

    cout << "Masukkan nama Anda: ";
    cin >> nama;

    tampilanSambutan(nama);
}
```

Prosedur tampilanSambutan menerima satu parameter bertipe string yang bernama nama. Di dalam prosedur, program menampilkan pesan sambutan dengan format "Selamat datang,

[nama]". Program meminta pengguna memasukkan nama mereka dan menyimpan input ke variabel nama. Setelah pengguna memasukkan nama, prosedur tampilkanSambutan(nama) dipanggil, yang menampilkan pesan sambutan dengan nama pengguna.

1.3 Deklarasi dan Pemanggilan Fungsi

Function atau fungsi adalah sub-program yang menerima data masukan dan memberikan/mengembalikan (return) sebuah nilai dari tipe tertentu (tipe dasar atau tipe bentukan). Fungsi memiliki struktur yang terdiri dari header, deklarasi dan badan fungsi (algoritma). Bagian header merupakan nama dari sebuah function, sedangkan deklarasi merupakan nama function yang hanya dipakai didalam function itu sendiri, dan badan fungsi merupakan sekumpulan instruksi-instruksi yang digunakan dalam function.

Sintaks:

```
TipeData NamaFungsi (DaftarParameter){  
    /*Code atau Badan Fungsi*/  
    return nilaireturn;  
}
```

Penjelasan:

- **TipeData**, Tipe data ini menentukan jenis nilai yang akan dikembalikan oleh fungsi (misalnya int, double, string, dll.). Tipe data ini juga menentukan jenis nilai yang harus digunakan di bagian return fungsi.
- **NamaFungsi**, Ini adalah nama fungsi yang digunakan sebagai identitas fungsi tersebut. Nama fungsi ini dipakai saat kita ingin memanggil atau menggunakan fungsi di bagian lain dari

program. Nama fungsi sebaiknya deskriptif agar mudah dipahami, misalnya hitungLuas, getUmur, dll.

- **DaftarParameter**, Parameter adalah variabel yang digunakan untuk menerima nilai yang diberikan saat fungsi dipanggil. Parameter ditempatkan di dalam tanda kurung (). Jika fungsi membutuhkan lebih dari satu parameter, maka parameter dipisahkan dengan koma (,). Fungsi dapat memiliki nol parameter (kosong), satu parameter, atau lebih dari satu parameter.
- **Badan Fungsi { ... }**, Blok { ... } adalah tubuh atau badan fungsi, yang berisi kode yang akan dieksekusi setiap kali fungsi dipanggil. Di dalam badan fungsi, kita menuliskan logika atau instruksi yang kita ingin jalankan.
- **Return**, return adalah perintah untuk mengembalikan nilai dari fungsi. nilaiReturn adalah nilai yang dikembalikan oleh fungsi sesuai dengan TipeData yang telah dideklarasikan. Setelah return dieksekusi, fungsi akan berhenti, dan nilai yang dikembalikan akan dikirim ke bagian program yang memanggil fungsi tersebut.

Pemanggilan fungsi dilakukan untuk menghasilkan nilai dengan melakukan pemanggilan nama fungsi dari program tersebut beserta parameternya, contohnya hitungLuas(10, 15).

Contoh:

Program 2. Fungsi Menghitung Luas

```
#include <iostream>
using namespace std;

int hitungLuas(int panjang, int lebar) {
    return panjang * lebar;
```

```
}

int main() {
    int panjang, lebar;

    cout << "Masukkan panjang: ";
    cin >> panjang;
    cout << "Masukkan lebar: ";
    cin >> lebar;

    int luas = hitungLuas(panjang, lebar);

    cout << "Luas persegi panjang adalah: " << luas
    << endl;

}
```

Fungsi hitungLuas menerima dua parameter (panjang dan lebar), menghitung luas, dan mengembalikan nilai luas tersebut ke main() menggunakan return. Di main(), setelah menerima input panjang dan lebar, fungsi hitungLuas(panjang, lebar) dipanggil, dan hasil perhitungan disimpan dalam variabel luas. Hasil kemudian ditampilkan di main().

1.4 Parameter dalam Prosedur dan Fungsi

Parameter dapat diartikan sebagai nama variabel yang dideklarasikan pada bagian header prosedur atau fungsi. Parameter yang disertakan ketika pemanggilan prosedur atau fungsi disebut dengan parameter actual (argument). Parameter yang dideklarasikan di dalam bagian header disebut parameter formal. Penggunaan parameter formal dilakukan dengan memanggil namanya dari program pemanggil disertakan parameter actual.

Parameter dalam pemrograman terbagi dua, yaitu

- **Parameter by Value:** Menyalin nilai argumen ke parameter fungsi. Setiap perubahan pada parameter tidak mempengaruhi nilai argumen asli di luar fungsi.
- **Parameter by Reference:** Menggunakan referensi (alamat) dari argumen. Perubahan pada parameter di dalam fungsi akan mempengaruhi nilai asli argumen di luar fungsi.

Contoh:

Program 3. Parameter

```
#include <iostream>
using namespace std;
void ubahByValue(int a) {
    a = a + 10;
}

void ubahByReference(int &b) {
    b = b + 10;
}

int main() {
    int x = 5, y = 5;

    ubahByValue(x);    // Parameter by value
    ubahByReference(y); // Parameter by Reference

    cout << "Nilai x (by value): " << x << endl; // Output: 5
    cout << "Nilai y (by reference): " << y << endl; // Output: 15
}
```

- **ubahByValue** menerima parameter by value, sehingga perubahan nilai a di dalam fungsi tidak mempengaruhi nilai x di luar fungsi.

- **ubahByReference** menerima parameter by reference, sehingga perubahan nilai b di dalam fungsi juga mengubah nilai y di luar fungsi.

1.5 Fungsi Rekursif

Fungsi Rekursif adalah fungsi yang memanggil dirinya sendiri secara langsung atau tidak langsung. Fungsi rekursif memecahkan masalah menjadi sub kecil dari masalah yang sama hingga mencapai kondisi dasar (base case) di mana pemanggilan fungsi berhenti. Teknik ini sering digunakan dalam pemrograman untuk memecahkan masalah yang dapat dipecah menjadi submasalah yang lebih sederhana dan serupa. Fungsi rekursif memiliki 2 unsur penting, yaitu:

- **Kondisi Dasar (Base Case)**, yaitu kondisi di mana rekursi akan berhenti. Kondisi dasar diperlukan agar fungsi tidak memanggil dirinya sendiri terus-menerus tanpa akhir.
- **Panggilan Rekursi**, jika kondisi dasar belum tercapai, fungsi akan memanggil dirinya sendiri dengan parameter yang dimodifikasi, mendekati kondisi dasar.

Contoh:

Program 4. Faktorial

```
#include <iostream>
using namespace std;

int faktorial(int n) {
    // Kondisi dasar
    if (n == 0 || n == 1) {
        return 1;
    }
    // Panggilan rekursif
    return n * faktorial(n - 1);
}
```

```
int main() {
    int angka;
    cout << "Masukkan angka: ";
    cin >> angka;

    cout << "Faktorial dari " << angka << " adalah
" << faktorial(angka) << endl;
    return 0;
}
```

- **Kondisi Dasar:** Jika n bernilai 0 atau 1, fungsi faktorial akan mengembalikan nilai 1. Ini menghentikan panggilan rekursif lebih lanjut.
- **Panggilan Rekursif:** Jika n lebih besar dari 1, fungsi akan memanggil dirinya sendiri dengan n - 1 dan mengalikan hasilnya dengan n. Hal ini terus berlanjut hingga mencapai kondisi dasar.

1.6 Pemrosesan Teks

String adalah tipe data yang digunakan untuk menyimpan urutan karakter, seperti kalimat atau kata. Dalam banyak bahasa pemrograman, string direpresentasikan sebagai array karakter atau objek yang menyediakan fungsi manipulasi teks. Di C++, string bisa dideklarasikan sebagai array karakter atau menggunakan kelas `std::string` dari library `string`, yang menyediakan cara lebih mudah dan aman untuk mengelola teks.

```
#include <string>
using namespace std;
string str = "Hello";
```

Pemrosesan teks atau string dalam C++ menawarkan berbagai operasi yang memudahkan manipulasi dan pengelolaan teks. Operasi-operasi dasar ini sangat berguna dalam berbagai aplikasi

pemrograman yang melibatkan teks. Terdapat banyak operasi dasar atau fungsi dalam pemrosesan teks atau string yaitu:

1) Menghitung panjang string

Menghitung panjang suatu variabel string dapat menggunakan metode `length()` atau `size()`.

```
string teks = "Hello World";
cout << "Panjang string: " <<
teks.length() << endl; // Output: 11
```

2) Menggabungkan string (concatenation)

Menyatukan dua atau lebih string menjadi satu string baru dapat menggunakan operator (+) atau metode `append(str)`.

```
string teks1 = "Hello";
string teks2 = " World";
string hasil = teks1 + teks2;
cout << hasil << endl; // Output: Hello
World
```

3) Mengakses karakter dalam string

Untuk mengakses suatu karakter dalam string dapat menggunakan index array atau metode `at(index)`.

```
string teks = "Hello";
cout << teks[1] << endl; // Output: e
cout << teks.at(1) << endl; // Output: e
```

4) Mengekstraksi substring

Ekstraksi substring yaitu mengambil bagian dari string, mulai dari posisi tertentu dengan panjang tertentu. Ekstraksi dapat

dilakukan menggunakan metode substr(pos, len). Dimana pos yaitu posisi awal substring dan len yaitu panjang substring.

```
string teks = "Hello World";
string bagian = teks.substr(6, 5);
cout << bagian << endl; // Output: World
```

5) Penggantian string

Mengganti sebagian dari string dengan string lain, mulai dari posisi tertentu dan sepanjang jumlah karakter tertentu dapat menggunakan metode replace(pos, len, new_substring). Dimana pos yaitu posisi awal penggantian, len yaitu panjang substring yang diganti, dan new_substring sebagai string baru.

```
string teks = "Hello World";
teks.replace(6, 5, "Everyone");
cout << teks << endl; // Output: Hello
Everyone
```

6) Konversi string

Mengkonversi tipe data lain (misalnya integer atau double) ke string menggunakan to_string(), atau sebaliknya menggunakan stoi(), stof(), stod().

```
int angka = 123;
string strAngka = to_string(angka);
cout << "Angka sebagai string: " <<
strAngka << endl;

string teks = "456";
int angka2 = stoi(teks);
cout << "String sebagai angka: " << angka2
<< endl;
```

7) Menghapus string

Menghapus bagian dari string mulai dari posisi tertentu sepanjang panjang tertentu menggunakan metode `erase(pos, len)`.

```
string teks = "Hello World";
teks.erase(5, 6); // Menghapus mulai dari
posisi 5 sebanyak 6 karakter (termasuk
spasi dan "World")
cout << teks << endl; // Output: Hello
```

2. Rangkuman

Prosedur dan fungsi adalah elemen dalam algoritma dan pemrograman yang berguna untuk mengorganisir kode menjadi lebih terstruktur, modular, dan mudah dikelola. **Prosedur** adalah fungsi tanpa nilai kembalian (`void`), sedangkan **fungsi** memiliki nilai kembalian. Fungsi rekursif adalah fungsi yang memanggil dirinya sendiri. Pemrosesan teks melibatkan operasi pada string, seperti:

- Menghitung Panjang String: Menggunakan `length()` atau `size()`.
- Menggabungkan String: Menggunakan operator `+` atau `append()`.
- Mengakses Karakter dalam String: Menggunakan operator `[]` atau `.at()`.
- Mengekstraksi Substring: Menggunakan `substr(pos, len)`.
- Penggantian String: Menggunakan `replace(pos, len, new_substring)`.
- Konversi String: Dari tipe lain ke string (`to_string()`) atau dari string ke tipe lain (`stoi()`, `stof()`, dll.).

- Menghapus String: Menggunakan `erase(pos, len)` untuk menghapus bagian string.

3. Pustaka

- [1] Munir, Rinaldi dan Lidya, Leony. 2016. Algoritma dan Pemrograman dalam Bahasa Pascal, C dan C++ Edisi Keenam. Informatika: Bandung.
- [2] Putri, M. P., Barovih, G., Azdy, R. A., Yuniansyah, Saputra, A., Sriyeni, Y., Rini, A., & Admojo, F. T. (2022). Algoritma dan Struktur Data. Bandung: Widina Bhakti Persada Bandung. ISBN 978-623-459-182-8.

D. Penilaian Kegiatan Belajar

Bagian ini terdiri atas bentuk penilaian kegiatan belajar yang disesuaikan dengan RPS serta umpan balik dan tindak lanjutnya.

1. Penilaian

1.1 Latihan

1) Konversi Suhu

Buat program untuk mengkonversi suhu dari derajat Celcius ke Fahrenheit dan Kelvin. Program harus meminta suhu dalam Celcius dari pengguna, kemudian menampilkan suhu yang setara dalam Fahrenheit dan Kelvin. Buat kedalam fungsi dan procedure, dimana:

Fungsi konversiKeFahrenheit untuk mengkonversi suhu ke fahrenheit.

Prosedur tampilanKelvin untuk menghitung suhu dalam Kelvin.

Gunakan rumus:

$$\bullet \text{Fahrenheit} = (\text{Celsius} * 9/5) + 32$$

$$\bullet \text{Kelvin} = \text{Celsius} + 273.15$$

1.2 Tantangan

1) Program Faktorial Ganjil Genap

Pak Dengklek yang sangat menyukai matematika sedang mengajarkan faktorial kepada Ucil, bebek Pak Dengklek. Karena Ucil sangatlah cerdas, ia pun dengan mudahnya mengerti faktorial. Oleh karena itu, Pak Dengklek

memberikan tantangan baru kepada Ucil untuk mempelajari faktorial ganjil-genap.

Perhitungan faktorial ganjil-genap sama seperti faktorial biasa, hanya saja semua bilangan genap yang dikalikan harus dibagi dengan 2 terlebih dahulu. Seperti kita ketahui bahwa notasi dari faktorial untuk bilangan bulat N adalah $N!$. Sedangkan, notasi dari faktorial ganjil-genap untuk N adalah $N!!$.

Perhitungan dari $N!$ adalah sebagai berikut:

$N!$

$$= N \times (N-1)!$$

$$= N \times (N-1) \times (N-2) \times \dots \times 1$$

Sedangkan, perhitungan dari $N!!$ adalah sebagai berikut:

$N!!$

$$= f(N) \times (N-1)!!$$

$$= f(N) \times f(N-1) \times f(N-2) \times \dots \times f(1)$$

dengan $f(x) = x$ apabila x ganjil, atau $x/2$ apabila x genap.

Sebagai contoh, $5!! = 5 \times 4/2 \times 3 \times 2/2 \times 1 = 30$.

Ucil diberikan sebuah bilangan bulat N. Bantulah Ucil membuatkan program untuk menghitung faktorial ganjil-genap dari N. Khusus untuk program ini, Ucil meminta agar Anda menggunakan rekursi.

2. Umpaman Balik dan Tindak Lanjut

1) Program Konversi Suhu

Program Konversi Suhu

```
#include <iostream>
using namespace std;

// Fungsi untuk mengonversi Celsius ke Fahrenheit
float konversiKeFahrenheit(float celsius) {
    return (celsius * 9/5) + 32;
}

// Prosedur untuk menampilkan hasil konversi ke
// Kelvin
void tampilanKelvin(float celsius) {
    float kelvin = celsius + 273.15;
    cout << "Suhu dalam Kelvin: " << kelvin << "
K" << endl;
}

int main() {
    float celsius;

    // Meminta input suhu dalam Celsius dari
    pengguna
    cout << "Masukkan suhu dalam Celsius: ";
    cin >> celsius;

    // Mengonversi suhu ke Fahrenheit
    float fahrenheit =
konversiKeFahrenheit(celsius);

    // Menampilkan hasil konversi
    cout << "Suhu dalam Fahrenheit: " << fahrenheit
<< " F" << endl;
    tampilanKelvin(celsius); // Menampilkan suhu
dalam Kelvin

    return 0;
}
```

ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA



MODUL VI

Array

**Elly Warni
Ingrid Nurtanio
Amil Ahmad Ilham
Adnan
Anugrayani Bustamin**

PRAKATA

Alhamdulillah, puji dan syukur kami panjatkan ke hadirat Allah SWT yang telah melimpahkan rahmat dan karunia-Nya sehingga modul ini dapat diselesaikan dengan baik. Shalawat dan salam senantiasa tercurah kepada Nabi Muhammad SAW, keluarga, dan para sahabat beliau yang telah membawa kita dari zaman kebodohan menuju terang benderang ilmu pengetahuan.

Modul ini disusun sebagai panduan pembelajaran untuk membantu mahasiswa memahami dan menguasai konsep dasar array dalam pemrograman. Array merupakan salah satu struktur data yang penting dan sering digunakan dalam pemrograman untuk mengelola data dalam jumlah besar secara efisien. Konsep ini sangat berguna dalam berbagai aplikasi pemrograman, terutama yang melibatkan pengolahan data secara intensif.

Kami menyadari bahwa penyusunan modul ini tidak lepas dari bantuan berbagai pihak. Oleh karena itu, kami mengucapkan terima kasih kepada semua pihak yang telah memberikan dukungan, masukan, dan saran dalam proses penyusunan modul ini. Ucapan terima kasih khusus kami sampaikan kepada rekan-rekan dosen dan staf yang telah memberikan kritik serta saran yang membangun.

Kami menyadari bahwa modul ini masih memiliki keterbatasan. Oleh karena itu, kami sangat mengharapkan kritik dan saran dari para pembaca untuk perbaikan di masa mendatang. Semoga modul ini dapat memberikan manfaat dan menjadi bahan referensi yang berguna bagi mahasiswa dalam memahami konsep array dan meningkatkan keterampilan pemrograman mereka.



Akhir kata, semoga upaya ini mendapat ridha dari Allah SWT dan dapat memberikan kontribusi positif dalam dunia pendidikan dan ilmu pengetahuan.

Makassar, 12 November 2022

Penulis

DAFTAR ISI

PRAKATA	i
DAFTAR ISI	iii
MODUL ALGORITMA DAN PEMROGRAMAN.....	1
Pengantar.....	1
KEGIATAN BELAJAR 1.....	3
Array – Larik.....	3
A. Deskripsi Singkat	3
B. Relevansi	4
C. Pembelajaran	5
D. Penilaian Kegiatan Belajar	16



UNIVERSITAS HASANUDDIN
FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK INFORMATIKA

Kode Dokumen

RENCANA PEMBELAJARAN SEMESTER

MATA KULIAH (MK)	KODE	Rumpun MK	BOBOT (sks)		SEMESTER	Tgl Penyusunan		
Algoritma Pemrograman dan Struktur Data	23D12110604	Teknik Informatika	T=2	P=2	2	19-02-2023		
OTORISASI	Pengembang RPS	Koordinator RMK		Ketua PRODI				
	Dr. Ir. Ingrid Nurtanio., MT Dr. Amil Ahmad Ilham., ST., M.IT Adnan., ST., MT., PhD Anugrayani Bustamin., ST., MT MT Elly Warni, S.T., M.T	Dr. Ir. Ingrid Nurtanio., MT		Prof. Dr. Indrabayu., ST., M.T., M.Bus.Sys				
Capaian Pembelajaran (CP)	CPL-PRODI yang dibebankan pada MK					<i>Intended Learning Outcomes</i>		
	CPL 1	Memiliki dasar pengetahuan Teknik Informatika yang meliputi teori dan konsep dasar dari Ilmu Komputer, Matematika dan Statistika, Algoritma dan Pemrograman, Rekayasa Perangkat Lunak, Manajemen Informasi dan Ketahanan Digital, serta pengetahuan tingkat lanjut pada bidang-bidang khusus Teknik Informatika, seperti Kecerdasan Buatan, Data Science, Jaringan Komputer, Komputasi Awan dan Internet of Things.						
	CPL 3	Mampu mengaplikasikan pengetahuan bidang Teknik Informatika yang dipadankan dengan bidang ilmu lainnya untuk menganalisa dan mencari solusi dari berbagai masalah berbasis komputasi.						
	CPL 4	Mampu mendesain, mengimplementasikan dan mengevaluasi solusi berbasis komputasi dengan mengaplikasikan ilmu Teknik Informatika dan dasar-dasar pembangunan perangkat lunak.						
	Capaian Pembelajaran Mata Kuliah (CPMK)							
	CPMK	Setelah mengikuti Mata Kuliah Algoritma Pemrograman dan Struktur Data selama 1 (satu) semester, mahasiswa mampu menguasai dan menerapkan konsep teoritis bidang Informatika dan ilmu komputer secara mendalam serta memformulasikan penyelesaian masalah procedural dengan pemikiran logis dan sistematis secara mandiri dan terukur sebagai dasar pengembangan perangkat lunak.						

	CPL ⇒ Sub-CPMK	
	CPL 1	Sub-CPMK 1 : Mahasiswa mampu mendefinisikan konsep dan pengertian Algoritma
	CPL 1	Sub-CPMK 2 : Mahasiswa mampu mendeskripsikan Dasar Algoritma
	CPL 3	Sub-CPMK 3 : Mahasiswa mampu mengimplementasikan Flowchart
	CPL 1	Sub-CPMK 4 : Mahasiswa mampu memahami bentuk pemilihan dan perulangan
	CPL 4	Sub-CPMK 5 : Mahasiswa mampu mengimplementasikan prosedur, fungsi dan pemrosesan teks
	CPL 1	Sub-CPMK 6 : Mahasiswa mampu mempraktekkan konsep array
	CPL 4	Sub-CPMK 7 : Mahasiswa mampu menetapkan ciri dari teknik pencarian
	CPL 4	Sub-CPMK 8 : Mahasiswa mampu menetapkan ciri dari teknik sorting dan menganalisis complexitas algoritma
	CPL 1	Sub-CPMK 9 : Mahasiswa mampu memahami struct dan konsep linked list
	CPL 1	Sub-CPMK 10 : Mahasiswa mampu memahami Antrian dan Stack
	CPL 1	Sub-CPMK 11 : Mahasiswa mampu memahami teknik graph
	CPL 3	Sub-CPMK 12 : Mahasiswa mampu mengimplementasi konsep Tree
Deskripsi Singkat MK	<p>Mata kuliah Algoritma Pemrograman dan Struktur Data merupakan mata kuliah yang menyajikan konsep pemrograman secara mendalam terkait algoritma dan struktur data. Muatan mata kuliah merupakan dasar dari ilmu matematika, algoritma dan logika yang menjadi syarat utama pada mata kuliah Pemrograman lanjut . Mata kuliah ini disajikan untuk mahasiswa semester dua di Program Studi Teknik Informatika, Penanggungjawab mata kuliah adalah Laboratorium Komputer . Metode Pembelajaran bauran, teori dan praktik</p>	
Bahan Kajian / Materi Pembelajaran	<ol style="list-style-type: none"> 1. Konsep Algoritma 2. Dasar Algoritma (Aturan Penulisan Header, deklarasi dan deskripsi) 3. Flowchart (Simbol, variabel, algoritma cabang, runtunan) 4. Pemilihan (Alir Kontrol dan Pengulangan) 5. Prosedur, Fungsi dan Pemrosesan Teks 6. Array – Larik 7. Pencarian (Binary Search) 8. Pencarian (Sequential Search) 9. Bubble dan Insertion Sort 10. Merge dan selection sort 11. Quick sort dan complexita algoritma 12. Struct dan Linked List 13. Antrian dan Stack 	

	14. Graph berarah dan tidak berarah 15. Tree								
Pustaka	Utama :	1. Munir, Rinaldi dan Lidya, Leony. 2016. Algoritma dan Pemrograman dalam Bahasa Pascal, C dan C++ Edisi Keenam. Informatika : Bandung 2. Gozali, William dan Aji, Alham Fikri. 2014. Pemrograman Kompetitif Dasar. Ikatan Alumni Tim Olimpiade Komputer Indonesia. 3. Zakaria, Teddy M dan Prijono, Agus. 2005. Konsep dan Implementasi Struktur Data. Informatika : Bandung. 4. Niswar, M., Ilham, A., Zainuddin, Z., Adnan, A., P, A., Warni, E., Aswad, I., & Muslimin, Z. (2021). Sosialisasi Metode Berfikir Komputasional pada Pendidikan Dasar dan Menengah di Lingkup Sulawesi Selatan. URNAL EPAT eknologi erapan ntuk engabdian asyarakat, 4(1), 46-52. https://doi.org/10.25042/jurnal_tepat.v4i1.172							
	Pendukung:	Tuliskan pustaka pendukung jika ada, sebagai pengayaan literasi							
Dosen Pengampu	1. Dr. Ir. Ingrid Nurtanio., MT 2. Dr. Amil Ahmad Ilham., ST., M.IT 3. Adnan., ST., MT., PhD 4. Anugrayani Bustamin., ST., MT 5. Elly Warni, S.T., M.T								
Matakuliah syarat	Dasar Pemrograman Komputer								
				Bentuk Pembelajaran,					
Pekan Ke-	Sub-CPMK (Kemampuan akhir tiap tahapan belajar)	Penilaian	Indikator	Metode Pembelajaran, Penugasan Mahasiswa, [Estimasi Waktu]		Materi Pembelajaran [Pustaka]	Bobot Penilaian (%)		
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)		

1	Mahasiswa mampu mengetahui konsep dan pengertian Algoritma	Mengetahui konsep definisi, manfaat, mekanisme dan kontribusi Algoritma	Assignments	Lecture TM: 4 x 50 menit.	Lecture VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Konsep dan definisi algoritma 2. Mekanisme dan Kontribusi Algoritma	5%
2	Mahasiswa mampu memahami Dasar Algoritma CPL 3- SubCPMK 2	Memahami Dasar Algoritma (tipe data, aturan penulisan, proses, instruksi dan aksi)	quizzes	Lecture, case study TM: 4 x 50 menit.	Lecture, case study VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Dasar Algoritma (tipe data, aturan penulisan, proses, instruksi, dan aksi) Pustaka : [1]	5%
3	Mahasiswa mampu mengenali Flowchart CPL 4- SubCPMK 3	Mampu mengenali Flowchart (simbol, variabel, algoritma cabang, dan runtunan)	Assignments	Lecture, small group discussion TM: 4 x 50 menit.	Lecture, small group discussion VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit	Flowchart (Simbol, variabel, algoritma cabang, runtunan) Pustaka : [1]	7%

4	Mahasiswa mampu memahami bentuk pemilihan (alih kontrol dan pengulangan)	Mampu memahami bentuk pemilihan (alih kontrol dan pengulangan)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Pemilihan Kontrol Pengulangan) Pustaka : [1]	(Alir dan 7%
5	Mahasiswa mampu mengimplementasikan prosedur, fungsi dan pemrosesan teks	Mampu mengimplementasikan pemanggilan prosedur, pemanggilan fungsi dan pemrosesan teks	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Pemanggilan Prosedur 2. Pemanggilan Fungsi 3. Pemrosesan Teks Pustaka : [1]	7%
6	Mahasiswa mampu mempraktekkan konsep array	Mampu mempraktekkan konsep array seperti menginisialisasi array, mengisi elemen, mencari nilai maksimum dan minimum	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Menginisialisasi array 2. Mengisi elemen array 3. Mencari nilai maksimum dan minimum Pustaka : [1][2]	7%

					x.php?id_session=13610		
7	Mahasiswa mampu menetapkan ciri dari teknik pencarian	Mampu menetapkan ciri dari teknik pencarian • Binary Search • Sequential Search	Assignments quizzes	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Binary Search 2. Sequential Search Pustaka : [2][3]	7%
8	Evaluasi Tengah Semester / Ujian Tengah Semester						
9	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting • Bubble Sort • Insertion Sort	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Bubble Sort 2. Insertion Sort Pustaka : [2][3]	9%
10	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting Merge Sort Selection Sort	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit	1. Merge Sort Selection Sort Pustaka : [2][3]	9%

					https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610		
11	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting Quick Sort berserta complexitas Algoritmanya	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Quick Sort 2. Complexitas Algoritma Pustaka : [2][3]	9%
12	Mahasiswa mampu memahami struct dan konsep linked list	Mampu memahami struct dan konsep dan operasi dasar linked list (deklarasi simpul, penambahan simpul pada linked list awal, akhir dan tengah)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Konsep Struct dan implementasinya 2. Deklarasi list 3. Penambahan simpul pada list awal, akhir dan tengah Pustaka : [2][3]	7%
13	Mahasiswa mampu memahami Antrian dan Stack	Mampu memahami Antrian dan Stack dalam List	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit	Antrian dan Stack dalam list Pustaka : [2][3]	7%

					https://sikola. https://sikola.un has.ac.id/courses /104D4224/inde x.php?id_session =13610		
14	Mahasiswa mampu memahami teknik graph	mampu memahami teknik graph (graph berarah dan tidak berarah menggunakan linked list)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.un has.ac.id/courses /104D 4224/index.php? id_session=1361 0	1. Graph berarah Graph tidak berarah Pustaka : [2][3]	7%
15	Mahasiswa mampu mengimplementasi konsep Tree	Mampu mengimplementasi konsep Tree menggunakan linked list	Assignments quizzes	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.un has.ac.id/courses /104D 4224/index.php? id_session=1361 0	Konsep tree menggunakan linked List Pustaka : [2][3]	7%
16	Evaluasi Akhir Semester / Ujian Akhir Semester						

MODUL ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA

MODUL VI

ARRAY - LARIK

Pengantar

Modul ini bertujuan untuk membantu mahasiswa memahami dan mempraktikkan konsep dasar array dalam pemrograman. Array adalah struktur data penting yang memungkinkan penyimpanan banyak data dalam satu variabel dengan indeks tertentu. Konsep array sangat berguna dalam berbagai aplikasi pemrograman, terutama yang melibatkan pengolahan data dalam jumlah besar dan pengaksesan elemen secara efisien.

Pada kegiatan belajar kali ini, mahasiswa akan mempelajari cara menginisialisasi array, mengisi elemen-elemen array, serta menggunakan array dua dimensi. Mahasiswa juga akan belajar bagaimana menggunakan indeks untuk mengakses dan memanipulasi data yang tersimpan dalam array. Pemahaman tentang array ini sangat penting karena array memungkinkan mahasiswa untuk mengelola data dalam skala besar dengan lebih terstruktur, efisien, dan sistematis.

Kegiatan ini mencakup perkuliahan tatap muka, diskusi kelompok, dan praktikum untuk memperdalam pemahaman mahasiswa mengenai penggunaan array dalam pemrograman. Mahasiswa diharapkan dapat menguasai materi ini melalui pengajaran selama 4x50 menit untuk tatap muka, serta kegiatan tambahan di virtual classroom (VC), praktikum terbimbing (PT), dan bimbingan mandiri (BM) masing-masing selama 4x60 menit.

Melalui modul ini, mahasiswa diharapkan mampu menginisialisasi array, mengisi elemen-elemen array, serta mencari nilai maksimum dan minimum dalam array yang mereka buat. Pemahaman tentang array ini akan memberikan dasar yang kuat bagi mahasiswa untuk memahami struktur data yang lebih kompleks pada modul-modul berikutnya.

KEGIATAN BELAJAR 1

Array – Larik

A. Deskripsi Singkat

Materi ini bertujuan untuk membantu mahasiswa menguasai konsep dasar array dalam pemrograman, yang merupakan lanjutan dari materi prosedur dan fungsi pada modul sebelumnya. Array adalah struktur data penting yang memungkinkan penyimpanan dan pengelolaan sejumlah elemen dengan tipe data yang sama dalam satu variabel, menggunakan indeks untuk mengakses setiap elemen. Pemahaman tentang array sangat penting dalam pemrograman, karena array memungkinkan pengolahan data dalam jumlah besar secara efisien, yang sering dibutuhkan dalam berbagai aplikasi.

Pada modul ini, mahasiswa akan belajar cara menginisialisasi array, yaitu menentukan jumlah elemen dan tipe data yang akan disimpan. Selain itu, mahasiswa akan mempelajari teknik untuk mengisi elemen-elemen dalam array baik secara manual maupun otomatis melalui perulangan. Hal ini penting untuk memahami bagaimana array dapat digunakan untuk menyimpan dan memproses data dalam aplikasi nyata.

Pemahaman yang mendalam tentang konsep array ini akan menjadi landasan penting bagi mahasiswa dalam mempelajari teknik pemrograman yang lebih kompleks pada modul berikutnya, seperti teknik pencarian dan pengurutan. Array memungkinkan mahasiswa untuk bekerja dengan data yang lebih besar dan lebih dinamis, serta memberi mereka fleksibilitas dalam menyusun logika program yang lebih efisien dan terstruktur.

B. Relevansi

Pada modul sebelumnya, mahasiswa mempelajari tentang prosedur dan fungsi yang membantu mereka dalam membuat program yang lebih modular dan terstruktur. Pemahaman tersebut memberi mahasiswa keterampilan untuk membagi program menjadi blok-blok kecil yang lebih mudah dikelola dan digunakan kembali. Pengetahuan ini sangat penting karena array sering digunakan bersama dengan prosedur dan fungsi untuk mengelola data yang lebih besar dan kompleks secara efisien.

Materi array dalam modul ini memungkinkan mahasiswa untuk menyimpan dan mengelola sejumlah besar data dalam satu variabel menggunakan indeks. Mahasiswa akan belajar bagaimana menginisialisasi, mengisi, dan mengakses elemen array. Keterampilan ini sangat penting untuk mengembangkan program yang lebih kompleks, terutama dalam aplikasi yang memerlukan pengolahan data dalam skala besar.

Pemahaman tentang array ini akan menjadi landasan penting bagi mahasiswa dalam mencapai Sub-CPMK berikutnya, yaitu teknik pencarian dan pengurutan. Dengan memahami dasar-dasar array, mahasiswa akan lebih siap untuk mempelajari teknik pemrograman yang lebih canggih yang melibatkan pengolahan dan manipulasi data secara efisien.

Secara keseluruhan, setelah mempelajari modul ini, mahasiswa diharapkan mampu mengimplementasikan array untuk menyimpan dan memproses data dalam program yang mereka buat. Kemampuan ini akan membantu mahasiswa merancang program yang lebih efisien dan mempersiapkan mereka untuk menguasai konsep-konsep pengolahan data yang lebih kompleks pada modul-modul berikutnya.

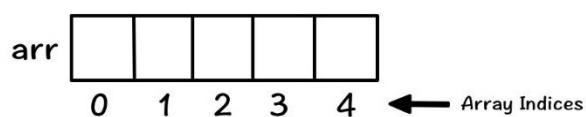
C. Pembelajaran

Capaian pembelajaran yang diharapkan pada kegiatan belajar yang pertama ini adalah peserta kuliah akan mampu memahami dan mengimplementasikan array dalam pemrograman. Untuk mencapai kompetensi lulusan ini, peserta kuliah diharapkan membaca dan menyelesaikan Kegiatan Belajar 1 ini. Untuk mencapai capaian pembelajaran pada Kegiatan Belajar 1, peserta kuliah harus mengikuti urutan kegiatan pembelajaran yang telah disiapkan dalam modul ini.

1. Uraian

1.1 Pendahuluan

Array adalah sebuah tipe data terstruktur yang dapat diterapkan pada suatu variabel yang dapat menyimpan banyak data dengan tipe sejenis atau homogen. Array berfungsi untuk mempermudah dalam penyimpanan data, sebagai contoh suatu data memiliki dua puluh nilai, jika tidak dideskripsikan menggunakan array, data tersebut haruslah dibuat menjadi dua puluh variabel. Belum lagi saat nanti data akan diakses maka kemungkinan data akan rumit dibaca karena variabel yang begitu banyak dengan nama yang berbeda. Oleh karena itu, digunakanlah array yang hanya dideklarasikan satu kali dan setiap nilai yang berbeda dapat disimpan pada indeks yang berbeda.



Gambar ini merupakan ilustrasi dasar array satu dimensi dengan lima elemen, yang diakses menggunakan indeks mulai dari 0 hingga 4. Array ini dapat menyimpan beberapa nilai



dengan tipe data yang sama dalam satu variabel, sehingga tidak perlu lagi mendeklarasikan banyak variabel.

1.2 Deklarasi dan Inisialisasi Array

Untuk mendeklarasikan array hal yang harus dilakukan terlebih dahulu adalah mendeklarasikan variabel dengan tipe data yang mengarah kepada array dan jumlah elemen array, dimana array tersebut adalah sebuah objek.

```
tipeData namaArray[jumlahElemen];
```

Contoh:

```
int angka[5];
```

Kode di atas menunjukkan bahwa telah dideklarasikan array dengan nama yaitu angka dan tipe data berupa integer. Lalu array tersebut memiliki jumlah elemen yaitu 5, dimana setiap elemen array haruslah berupa integer.

Inisialisasi array dapat dilakukan dengan tiga cara, yaitu:

- Inisialisasi saat deklarasi

Contoh:

```
int angka[5] = {1, 2, 3, 4, 5};
```

- Inisialisasi tanpa menentukan ukuran

Contoh:

```
int angka[] = {1, 2, 3, 4, 5};
```

- Inisialisasi setelah deklarasi

```
int angka[2];
```

```
angka[0] = 10;
```

```
angka[1] = 20;
```

Array hanya dapat berisi sesuai dengan jumlah kapasitas yang sudah dideklarasikan, jika indeks array yang diakses lebih dari kapasitasnya maka akan terjadi error pada program yang dijalankan karena elemen yang di input belum ada di dalam deklarasi elemen array.

1.3 Mengakses Array

Untuk mengakses array yang sudah dideklarasikan sebelumnya dapat dilakukan dengan menggunakan operator subskrip array ([]). Setiap elemen dalam array memiliki indeks yang dimulai dari 0 hingga (ukuran array – 1). Misalnya, angka[0] adalah elemen pertama, angka[1] adalah elemen kedua, dan seterusnya.

Contoh:

```
int angka[5] = {1, 2, 3, 4, 5};  
cout << angka[2]; // Menampilkan elemen ketiga, yaitu 3
```

Selain itu, juga dapat menggunakan looping untuk mengakses setiap elemen array. Ini memudahkan untuk melakukan operasi yang sama pada semua elemen.

Contoh:

```
for (int i = 0; i < 5; i++) {  
    cout << angka[i] << " ";  
}
```

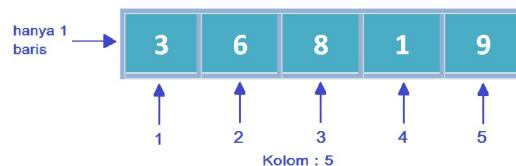
Atau

```
for (int i : angka) {  
    cout << i << " ";  
}
```

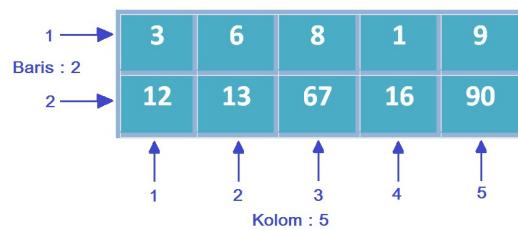
1.4 Array Multi Dimensi

Array multi dimensi adalah array yang memiliki lebih dari satu indeks untuk mengakses setiap elemen, memungkinkan penyimpanan data dalam bentuk matriks atau tabel. Array multi dimensi biasanya digunakan untuk representasi data yang memiliki lebih dari satu atribut (baris dan kolom) atau data yang membutuhkan organisasi lebih kompleks, seperti tabel atau grafik 3D.

Array 1 Dimensi



Array 2 Dimensi



Untuk deklarasi, inisialisasi, dan mengaksesnya sama dengan array 1 dimensi. Adapun penjelasan lengkapnya yaitu:

- 
- 1) Deklarasi, Array multi dimensi dideklarasikan dengan menentukan jumlah tiap dimensinya.

Contoh:

```
int matriks[3][4]; // Array 2D dengan 3 baris dan 4 kolom
```

- 2) Inisialisasi, Kita dapat menginisialisasi array dua dimensi saat deklarasi dengan nilai awal.

Contoh:

```
int matriks[3][4] = {  
    {1, 2, 3, 4},  
    {5, 6, 7, 8},  
    {9, 10, 11, 12}  
};
```

Ini membuat array matriks dengan 3 baris dan 4 kolom, dan setiap elemen sudah diisi dengan nilai tertentu.

3) Mengakses Array

Untuk mengakses elemen dalam array dua dimensi, kita dapat menggunakan indeks tiap dimensinya, misal dua dimensi maka indeks baris dan kolom. Misalnya, matriks[1][2] mengakses elemen pada baris kedua dan kolom ketiga dari array matriks. Selain itu, juga dapat diakses menggunakan looping bersarang.

Contoh:

- akses indeks

```
cout << matriks[1][2]; // Output: 7
```

- looping bersarang

```
for (int i = 0; i < 3; i++) {  
  
    for (int j = 0; j < 4; j++) {  
  
        cout << matriks[i][j] << " ";  
  
    }  
  
    cout << endl;  
  
}
```

1.5 Array Dinamis

Array statis adalah array yang memiliki ukuran tetap. Ini berarti bahwa setelah kita menentukan ukuran array, kita tidak dapat mengubahnya selama program dijalankan. Nah, terdapat kasus dimana kita membutuhkan struktur data array yang dapat menambahkan dan menyisipkan elemen di dalamnya. Disinilah array dinamis digunakan. Array dinamis adalah array yang tidak memiliki ukuran tetap, artinya kita dapat mengubah ukuran array kapan saja selama program dijalankan. Pada bahasa C++, kita dapat menggunakan array dinamis menggunakan **std::vector**.

```
vector <tipedata> namavektor;
```

Dengan menggunakan vector, kita dapat menambahkan elemen baru pada vector dan juga melakukan beberapa operasi. Pada **std::vector**, telah diberikan banyak metode atau function yang dapat digunakan untuk memanipulasi atau mengakses informasi dari vector yang telah kita deklarasikan.

- **push_back()**: Menambahkan elemen di akhir vector.
- **pop_back()**: Menghapus elemen terakhir vector.

- **insert()**: Menyisipkan elemen pada posisi tertentu.
- **erase()**: Menghapus elemen pada posisi tertentu atau rentang elemen.
- **clear()**: Menghapus semua elemen dalam vector, mengosongkannya.

1.6 Prefix Sum

Array prefix sum adalah struktur data yang memungkinkan kita untuk menghitung jumlah elemen dalam rentang yang diberikan secara efisien. Array prefix sum menyimpan jumlah kumulatif elemen-elemen dalam array asli, dari awal hingga setiap indeks. Setelah kita membangun array prefix sum, kita dapat dengan cepat menjawab pertanyaan tentang jumlah elemen dalam rentang mana pun dalam waktu yang konstan, **O(1)**.

Misalkan kita memiliki array asli `arr` dengan n elemen. Array prefix sum `prefix_sum` didefinisikan sebagai:

- `prefix_sum[i] = jumlah elemen arr[0] hingga arr[i]`

Jadi, elemen `prefix_sum[i]` menyimpan jumlah semua elemen dari awal hingga indeks i dalam array `arr`.

Dengan menggunakan array prefix sum ini, kita dapat menghitung jumlah elemen dalam rentang $[L, R]$ menggunakan rumus berikut:

- $\text{sum}(L, R) = \text{prefix_sum}[R] - \text{prefix_sum}[L - 1]$
jika $L > 0$
- $\text{sum}(0, R) = \text{prefix_sum}[R]$ jika $L = 0$

1.7 Contoh Penggunaan Array

1) Array dua dimensi

Misalkan kita ingin menyimpan nilai dari N mahasiswa dalam M mata pelajaran. Dalam hal ini, kita akan menggunakan **array 2 dimensi** untuk menyimpan nilai setiap mahasiswa pada masing-masing mata pelajaran.

	Matkul 1	Matkul 2	Matkul 3
Mahasiswa 1	80	75	60
Mahasiswa 2	75	90	85

Source code:

Program 1. Input Nilai Mahasiswa

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    const int N = 2; // Jumlah mahasiswa
    const int M = 3; // Jumlah mata pelajaran

    // Deklarasi array 2D untuk menyimpan nilai
    int nilai[N][M];

    // Input nilai mahasiswa
    cout << "Masukkan nilai mahasiswa untuk
setiap mata pelajaran:\n";
    for (int i = 0; i < N; i++) {
        cout << "Mahasiswa " << (i + 1) << ":\n";
        for (int j = 0; j < M; j++) {
            cout << " Mata Pelajaran " << (j +
1) << ": ";
            cin >> nilai[i][j];
        }
    }
}
```

```
    }  
  
    return 0;  
}
```

Output:

```
Mahasiswa 1:  
Mata Pelajaran 1: 80  
Mata Pelajaran 2: 75  
Mata Pelajaran 3: 60  
Mahasiswa 2:  
Mata Pelajaran 1: 75  
Mata Pelajaran 2: 90  
Mata Pelajaran 3: 85
```

2) Prefix sum

Misalkan kita memiliki sebuah array yang menyimpan angka-angka, dan kita ingin menjawab beberapa permintaan (query) yang meminta jumlah elemen dari indeks L hingga R dalam array tersebut. Dengan menggunakan prefix sum, kita dapat menghitung jumlah elemen dalam rentang tersebut dengan cepat.

Source Code:

Program 2. Prefix Sum Input Nilai Mahasiswa

```
#include <bits/stdc++.h>  
using namespace std;  
  
int main() {  
    vector<int> arr = {3, 1, 4, 1, 5, 9, 2, 6,  
5, 3};  
    int n = arr.size();  
  
    // Membuat array prefix sum  
    vector<int> prefixSum(n);  
    prefixSum[0] = arr[0];
```

```
// Menghitung prefix sum
for (int i = 1; i < n; i++) {
    prefixSum[i] = prefixSum[i - 1] + arr[i];
}

// Jumlah elemen dari indeks L hingga R
int L, R;
cout << "Masukkan rentang indeks (L R): ";
cin >> L >> R;

// Menghitung jumlah dalam rentang [L, R]
int sum;
if (L == 0) {
    sum = prefixSum[R];
} else {
    sum = prefixSum[R] - prefixSum[L - 1];
}

cout << "Jumlah elemen dari indeks " << L <<
" hingga " << R << " adalah: " << sum << endl;

}
```

Output:

```
Masukkan rentang indeks (L R): 2 8
Jumlah elemen dari indeks 2 hingga 8 adalah: 32
```

2. Rangkuman

Array adalah tipe data terstruktur yang memungkinkan penyimpanan banyak data dengan tipe yang sama dalam satu variabel. Deklarasi array memerlukan tipe data, nama array, dan jumlah elemen yang dibutuhkan. Inisialisasi array dapat dilakukan langsung saat deklarasi atau setelah deklarasi dengan menetapkan nilai per elemen menggunakan indeks. Akses ke elemen array

dilakukan menggunakan indeks, yang dimulai dari 0. Untuk mengakses seluruh elemen array, kita dapat menggunakan loop, seperti loop `for` yang memungkinkan pengulangan akses pada setiap elemen array.

Terdapat array multidimensi yang memungkinkan penyimpanan data dalam bentuk matriks atau tabel, dengan lebih dari satu indeks untuk mengakses data. C++ menyediakan array dinamis melalui `std::vector`, yang memungkinkan penambahan dan penghapusan elemen secara dinamis. Library `<vector>` di C++ menawarkan berbagai fungsi untuk memanipulasi array dinamis, seperti `push_back()` untuk menambahkan elemen di akhir vector dan `pop_back()` untuk menghapus elemen terakhir.

Teknik lain yang berguna dalam manipulasi array adalah prefix sum, yang digunakan untuk menghitung jumlah elemen dalam rentang tertentu dengan cepat. Array prefix sum menyimpan jumlah kumulatif dari elemen-elemen dalam array asli hingga indeks tertentu, sehingga pertanyaan mengenai jumlah elemen dalam suatu rentang dapat dijawab dalam waktu konstan ($O(1)$).

3. Pustaka

- [1] Munir, Rinaldi dan Lidya, Leony. 2016. Algoritma dan Pemrograman dalam Bahasa Pascal, C dan C++ Edisi Keenam. Informatika: Bandung.
- [2] Putri, M. P., Barovih, G., Azdy, R. A., Yuniansyah, Saputra, A., Sriyeni, Y., Rini, A., & Admojo, F. T. (2022). Algoritma dan Struktur Data. Bandung: Widina Bhakti Persada Bandung. ISBN 978-623-459-182-8.

D. Penilaian Kegiatan Belajar

Bagian ini terdiri atas bentuk penilaian kegiatan belajar yang disesuaikan dengan RPS serta umpan balik dan tindak lanjutnya.

1. Penilaian

1.1 Latihan

1) Menghitung Jumlah Elemen Genap dan Ganjil dalam Array

Diberikan sebuah array yang berisi N bilangan bulat. Elemen array tersebut akan diinputkan oleh user. Hitung berapa banyak elemen yang bernilai genap dan berapa banyak yang bernilai ganjil.

Input:

- Integer N yang menunjukkan ukuran array.
- N bilangan bulat sebagai elemen array.

Output:

- Dua angka, jumlah elemen genap dan jumlah elemen ganjil dalam array.

1.2 Tantangan

1) Lumpia Rebung

Deskripsi Masalah

Gema dan Astik sedang mengikuti lomba pemrograman di Semarang. Sebelum pulang, Gema membelikan Astik oleh-oleh berupa lumpia rebung sepanjang L cm.

Karena Astik mempunyai N teman, maka Astik ingin memotong lumpia menjadi $N + 1$ bagian (1 bagian untuk dirinya sendiri) dengan memotong N kali. Pada mulanya,

lumpia tersebut terdiri atas 1 bagian sepanjang L cm. Pada pemotongan ke- i , Astik memotong bagian ke- B_i dari kiri menjadi 2 bagian sama panjang. Dijamin bagian tersebut mempunyai panjang bilangan genap dalam satuan cm.

Kini Gema penasaran, berapa panjang masing-masing bagian setelah dipotong oleh Astik. Bantulah Gema menghitungnya!

Format Masukan dan Keluaran

Baris pertama masukan terdiri dari dua buah bilangan N dan L yang menyatakan banyaknya teman Astik dan panjang lumpia rebung dalam cm. Baris kedua terdiri dari N buah bilangan yang menyatakan nilai B_1, B_2, \dots, B_N ($1 \leq B_i \leq i$).

Keluaran terdiri dari sebuah baris berisi $N + 1$ bilangan yang menyatakan panjang masing-masing bagian dari kiri sampai kanan dalam cm.

Contoh Masukan/Keluaran

Masukan	Keluaran
4 1000 1 2 2 1	250 250 125 125 250

Penjelasan:

- Pada mulanya, lumpia terdiri dari 1 bagian sepanjang 1000 cm.
- Setelah dipotong untuk pertama kalinya, lumpia terdiri dari 2 bagian sepanjang 500 cm dan 500 cm.

- Selanjutnya bagian kedua dari kiri dipotong, lumpia terdiri dari 3 bagian sepanjang 500 cm, 250 cm, dan 250 cm.
- Kemudian bagian kedua dari kiri dipotong, lumpia terdiri dari 4 bagian sepanjang 500 cm, 125 cm, 125 cm, dan 250 cm.
- Terakhir, bagian terkiri dipotong sehingga lumpia terdiri dari 5 bagian sepanjang 250 cm, 250 cm, 125 cm, 125 cm, dan 250 cm.

2. Umpam Balik dan Tindak Lanjut

1) Program Menghitung Jumlah Elemen Genap dan Ganjil

Program 2. Prefix Sum Input Nilai Mahasiswa

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    vector<int> arr = {3, 1, 4, 1, 5, 9, 2, 6,
5, 3};
    int n = arr.size();

    // Membuat array prefix sum
    vector<int> prefixSum(n);
    prefixSum[0] = arr[0];

    // Menghitung prefix sum
    for (int i = 1; i < n; i++) {
        prefixSum[i] = prefixSum[i - 1] + arr[i];
    }

    // Jumlah elemen dari indeks L hingga R
    int L, R;
    cout << "Masukkan rentang indeks (L R): ";
    cin >> L >> R;

    // Menghitung jumlah dalam rentang [L, R]
```

```
int sum;
if (L == 0) {
    sum = prefixSum[R];
} else {
    sum = prefixSum[R] - prefixSum[L - 1];
}

cout << "Jumlah elemen dari indeks " << L <<
" hingga " << R << " adalah: " << sum << endl;
```



UNIVERSITAS HASANUDDIN

ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA

MODUL VII

Searching



Elly Warni
Ingrid Nurtanio
Amil Ahmad Ilham
Adnan
Anugrayani Bustamin

PRAKATA

Alhamdulillah, puji dan syukur kami panjatkan ke hadirat Allah SWT yang telah melimpahkan rahmat dan hidayah-Nya, sehingga modul ini dapat terselesaikan dengan baik. Shalawat serta salam senantiasa tercurah kepada junjungan kita, Nabi Muhammad SAW, beserta keluarga dan sahabat beliau yang telah membawa kita dari zaman kegelapan menuju terang benderang ilmu pengetahuan.

Modul ini disusun sebagai bagian dari rangkaian pembelajaran yang bertujuan untuk membantu mahasiswa memahami algoritma searching. Modul ini merupakan modul lanjutan dari modul sebelumnya yang membahas dasar algoritma dan beberapa fungsi pada algoritma pemrograman, dimana pada modul ini mahasiswa mulai menerapkan hal tersebut pada pemahaman konsep serta implementasi algoritma pencarian dalam bahasa pemrograman.

Tidak dapat dipungkiri bahwa proses penyusunan modul ini tidak lepas dari dukungan berbagai pihak. Oleh karena itu, kami mengucapkan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan masukan dan bimbingan dalam proses penyusunan modul ini. Terima kasih kami sampaikan kepada rekan-rekan dosen dan staf yang telah memberikan kritik dan saran yang membangun, sehingga modul ini dapat terselesaikan dengan baik.

Kami menyadari bahwa modul ini masih memiliki keterbatasan dan kekurangan. Oleh karena itu, kami sangat mengharapkan masukan, kritik, dan saran dari para pembaca untuk penyempurnaan di masa yang akan datang. Semoga modul ini dapat memberikan manfaat dan menjadi bahan referensi yang berharga bagi para mahasiswa dalam memahami konsep algoritma dan pemrograman.



Akhir kata, semoga upaya ini mendapat ridha Allah SWT dan memberikan kontribusi positif bagi kemajuan ilmu pengetahuan. Amin.

Makassar, November 2024

Penulis

DAFTAR ISI

PRAKATA	i
DAFTAR ISI	iii
RPS MATA KULIAH ALGORITMA DAN STRUKTUR DATA.....	1
MODUL ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA.....	12
Pengantar.....	12
KEGIATAN BELAJAR 1.....	14
A. Deskripsi Singkat	14
B. Relevansi.....	14
C. Pembelajaran.....	15
D. Penilaian Kegiatan Belajar	24

RPS MATA KULIAH ALGORITMA DAN STRUKTUR DATA

	UNIVERSITAS HASANUDDIN FAKULTAS TEKNIK PROGRAM STUDI TEKNIK INFORMATIKA					Kode Dokumen
RENCANA PEMBELAJARAN SEMESTER						
MATA KULIAH (MK)	KODE	Rumpun MK	BOBOT (skn)		SEMESTER	Tgl Penyusunan
Algoritma Pemrograman dan Struktur Data	23D12110604	Teknik Informatika	T=2	P=2	2	19-02-2023
OTORISASI	Pengembang RPS	Koordinator RMK		Ketua PRODI		
	Dr. Ir. Ingrid Nurtanio., MT Dr. Amil Ahmad Ilham., ST., M.IT Adnan., ST., MT., PhD Anugrayani Bustamin., ST., MT Elly Warni, S.T., M.T	Dr. Ir. Ingrid Nurtanio., MT		Prof. Dr. Indrabayu., ST., M.T., M.Bus.Sys		
	CPL-PRODI yang dibebankan pada MK <i>Intended Learning Outcomes</i>					

Capaian Pembelajaran (CP)	CPL 1	Memiliki dasar pengetahuan Teknik Informatika yang meliputi teori dan konsep dasar dari Ilmu Komputer, Matematika dan Statistika, Algoritma dan Pemrograman, Rekayasa Perangkat Lunak, Manajemen Informasi dan Ketahanan Digital, serta pengetahuan tingkat lanjut pada bidang-bidang khusus Teknik Informatika, seperti Kecerdasan Buatan, Data Science, Jaringan Komputer, Komputasi Awan dan Internet of Things.
	CPL 3	Mampu mengaplikasikan pengetahuan bidang Teknik Informatika yang dipadankan dengan bidang ilmu lainnya untuk menganalisa dan mencari solusi dari berbagai masalah berbasis komputasi.
	CPL 4	Mampu mendesain, mengimplementasikan dan mengevaluasi solusi berbasis komputasi dengan mengaplikasikan ilmu Teknik Informatika dan dasar-dasar pembangunan perangkat lunak.
	Capaian Pembelajaran Mata Kuliah (CPMK)	
	CPMK	Setelah mengikuti Mata Kuliah Algoritma Pemrograman dan Struktur Data selama 1 (satu) semester, mahasiswa mampu menguasai dan menerapkan konsep teoritis bidang Informatika dan ilmu komputer secara dan mendalam serta memformulasikan penyelesaian masalah procedural dengan pemikiran logis dan sistematis secara mandiri dan terukur sebagai dasar pengembangan perangkat lunak.

	CPL ⇒ Sub-CPMK	
	CPL 1	Sub-CPMK 1 : Mahasiswa mampu mendefinisikan konsep dan pengertian Algoritma
	CPL 1	Sub-CPMK 2 : Mahasiswa mampu mendeksripsikan Dasar Algoritma
	CPL 3	Sub-CPMK 3 : Mahasiswa mampu mengimplementasikan Flowchart
	CPL 1	Sub-CPMK 4 : Mahasiswa mampu memahami bentuk pemilihan dan perulangan
	CPL 4	Sub-CPMK 5 : Mahasiswa mampu mengimplementasikan prosedur, fungsi dan pemrosesan teks
	CPL 1	Sub-CPMK 6 : Mahasiswa mampu mempraktekkan konsep array
	CPL 4	Sub-CPMK 7 : Mahasiswa mampu menetapkan ciri dari teknik pencarian
	CPL 4	Sub-CPMK 8 : Mahasiswa mampu menetapkan ciri dari teknik sorting dan menganalisis complexitas algoritma
	CPL 1	Sub-CPMK 9 : Mahasiswa mampu memahami struct dan konsep linked list
	CPL 1	Sub-CPMK 10 : Mahasiswa mampu memahami Antrian dan Stack
	CPL 1	Sub-CPMK 11 : Mahasiswa mampu memahami teknik graph
	CPL 3	Sub-CPMK 12 : Mahasiswa mampu mengimplementasi konsep Tree
Deskripsi Singkat MK	<p>Mata kuliah Algoritma Pemrograman dan Struktur Data merupakan mata kuliah yang menyajikan konsep pemrograman secara mendalam terkait algoritma dan struktur data. Muatan mata kuliah merupakan dasar dari ilmu matematika, algoritma dan logika yang menjadi syarat utama pada mata kuliah Pemrograman lanjut . Mata kuliah ini disajikan untuk mahasiswa semester dua di Program Studi Teknik Informatika, Penanggungjawab mata kuliah adalah Laboratorium Komputer . Metode Pembelajaran bauran, teori dan praktik</p>	
Bahan Kajian / Materi Pembelajaran	<ol style="list-style-type: none"> 1. Konsep Algoritma 2. Dasar Algoritma (Aturan Penulisan Header, deklarasi dan deskripsi) 3. Flowchart (Simbol, variabel, algoritma cabang, runtunan) 4. Pemilihan (Alir Kontrol dan Pengulangan) 5. Prosedur, Fungsi dan Pemrosesan Teks 6. Array – Larik 7. Pencarian (Binary Search) 	

	<ul style="list-style-type: none"> 8. Pencarian (Sequential Search) 9. Bubble dan Insertion Sort 10. Merge dan selection sort 11. Quick sort dan complexita algoritma 12. Struct dan Linked List 13. Antrian dan Stack 14. Graph berarah dan tidak berarah 15. Tree
--	---

Pustaka	Utama :	
	<ul style="list-style-type: none"> 1. Munir, Rinaldi dan Lidya, Leony. 2016. Algoritma dan Pemrograman dalam Bahasa Pascal, C dan C++ Edisi Keenam. Informatika : Bandung 2. Gozali, William dan Aji, Alham Fikri. 2014. Pemrograman Kompetitif Dasar. Ikatan Alumni Tim Olimpiade Komputer Indonesia. 3. Zakaria, Teddy M dan Prijono, Agus. 2005. Konsep dan Implementasii Struktur Data. Informatika : Bandung. 4. Niswar, M., Ilham, A., Zainuddin, Z., Adnan, A., P, A., Warni, E., Aswad, I., & Muslimin, Z. (2021). Sosialisasi Metode Berfikir Komputasional pada Pendidikan Dasar dan Menengah di Lingkup Sulawesi Selatan. URNAL EPAT eknologi erapan ntuk engabdian asyarakat, 4(1), 46-52. https://doi.org/10.25042/jurnal_tepat.v4i1.172 	
	Pendukung:	
	Tuliskan pustaka pendukung jika ada, sebagai pengayaan literasi	

Dosen Pengampu		1. Dr. Ir. Ingrid Nurtanio., MT 2. Dr. Amil Ahmad Ilham., ST., M.IT 3. Adnan., ST., MT., PhD 4. Anugrayani Bustamin., ST., MT 5. Elly Warni, S.T., M.T						
Matakuliah syarat		Dasar Pemrograman Komputer						
Pekan Ke-	Sub-CPMK (Kemampuan akhir tiap tahapan belajar)	Penilaian		Metode Pembelajaran, Penugasan Mahasiswa, [Estimasi Waktu]			Materi Pembelajaran [Pustaka]	Bobot Penilaian (%)
		Indikator	Kriteria & Bentuk	Luring (offline)	Daring (online)			
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	
1	Mahasiswa mampu mengetahui konsep dan pengertian Algoritma	Mengetahui konsep definisi, manfaat, mekanisme dan kontribusi Algoritma	Assignments	Lecture TM: 4 x 50 menit.	Lecture VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Konsep dan definisi algoritma 2. Mekanisme dan Kontribusi Algoritma	5%	

2	Mahasiswa mampu memahami Dasar Algoritma CPL 3- SubCPMK 2	Memahami Dasar Algoritma (tipe data, aturan penulisan, proses, instruksi dan aksi)	quizzes	Lecture, case study TM: 4 x 50 menit.	Lecture, case study VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Dasar Algoritma (tipe data, aturan penulisan, proses, instruksi, dan aksi) Pustaka : [1]	5%
3	Mahasiswa mampu mengenali Flowchart CPL 4- SubCPMK 3	Mampu mengenali Flowchart (simbol, variabel, algoritma cabang, dan runtunan)	Assignments	Lecture, small group discussion TM: 4 x 50 menit.	Lecture, small group discussion VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit	Flowchart (Simbol, variabel, algoritma cabang, runtunan) Pustaka : [1]	7%
4	Mahasiswa mampu memahami bentuk pemilihan (alih kontrol dan pengulangan)	Mampu memahami bentuk pemilihan (alih kontrol dan pengulangan)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses	Pemilihan Kontrol (Alir dan Pengulangan) Pustaka : [1]	7%

					/104D4224/inde x.php?id_session =13610		
5	Mahasiswa mampu mengimplementasikan prosedur, fungsi dan pemrosesan teks	Mampu mengimplementasikan pemanggilan prosedur, pemanggilan fungsi dan pemrosesan teks	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Pemanggilan Prosedur 2. Pemanggilan Fungsi 3. Pemrosesan Teks Pustaka : [1]	7%
6	Mahasiswa mampu mempraktekkan konsep array	Mampu mempraktekkan konsep array seperti menginisialisasi array, mengisi elemen, mencari nilai maksimum dan minimum	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Menginisialisasi array 2. Mengisi elemen array 3. Mencari nilai maksimum dan minimum Pustaka : [1][2]	7%

7	Mahasiswa mampu menetapkan ciri dari teknik pencarian	Mampu menetapkan ciri dari teknik pencarian • Binary Search • Sequential Search	Assignments quizzes	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Binary Search 2. Sequential Search Pustaka : [2][3]	7%
8	Evaluasi Tengah Semester / Ujian Tengah Semester						
9	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting • Bubble Sort • Insertion Sort	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Bubble Sort 2. Insertion Sort Pustaka : [2][3]	9%
	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting	Assignments	Lecture, practicum	Lecture, practicum VC: 4 x 50 menit.	1. Merge Sort Selection Sort Pustaka : [2][3]	9%

10		Merge Sort Selection Sort		TM: 4 x 50 menit.	PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610		
11	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting Quick Sort berserta complexitas Algoritmanya	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Quick Sort 2. Complexitas Algoritma Pustaka : [2][3]	9%
12	Mahasiswa mampu memahami struct dan konsep linked list	Mampu memahami struct dan konsep dan operasi dasar linked list (deklarasi simpul, penambahan)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit	1. Konsep Struct dan implementasinya 2. Deklarasi list 3. Penambahan simpul pada list awal, akhir dan tengah	7%

		(simpul pada linked list awal, akhir dan tengah)			https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Pustaka : [2][3]	
13	Mahasiswa mampu memahami Antrian dan Stack	Mampu memahami Antrian dan Stack dalam List	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Antrian dan Stack dalam list Pustaka : [2][3]	7%
14	Mahasiswa mampu memahami teknik graph	mampu memahami teknik graph (graph berarah dan tidak berarah menggunakan linked list)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Graph berarah Graph tidak berarah Pustaka : [2][3]	7%

					4224/index.php? id_session=1361 0		
15	Mahasiswa mampu mengimplementasi konsep Tree	Mampu mengimplementasi konsep Tree menggunakan linked list	Assignments quizzes	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Konsep tree menggunakan linked List Pustaka : [2][3]	7%
16	Evaluasi Akhir Semester / Ujian Akhir Semester						

MODUL ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA

MODUL VII

SEARCHING

Pengantar

Modul ini dirancang untuk membantu mahasiswa memahami dan mengenali ciri-ciri dari berbagai teknik pencarian dalam algoritma, khususnya pencarian binary search dan sequential search. Modul ini merupakan langkah penting untuk mencapai capaian pembelajaran mata kuliah (CPMK) yang telah ditetapkan, di mana mahasiswa diharapkan dapat mengenali dan membedakan berbagai metode pencarian serta memahami situasi yang tepat untuk menggunakannya.

Pada Sub-CPMK ini, mahasiswa diharapkan mampu memahami konsep dan karakteristik dari binary search dan sequential search sebagai teknik dasar pencarian data dalam struktur yang terurut maupun tidak terurut. Dengan mempelajari ciri-ciri binary search, mahasiswa akan memahami bagaimana pencarian data dapat dilakukan secara cepat dan efisien dalam data yang terurut. Sedangkan dengan memahami sequential search, mahasiswa akan mengenali bagaimana pencarian data dilakukan secara berurutan dalam data yang tidak terurut. Kedua metode ini akan memberikan wawasan yang mendalam tentang cara memilih metode pencarian yang paling sesuai berdasarkan jenis data dan konteks penggunaannya.

Kegiatan pembelajaran dalam modul ini akan dilakukan melalui diskusi kelompok kecil dan pembelajaran interaktif, didukung oleh platform pembelajaran daring. Mahasiswa akan mempelajari cara kerja kedua

metode pencarian ini, baik secara teoritis maupun dalam bentuk penerapan menggunakan pseudocode. Dengan pemahaman ini, mahasiswa akan memiliki keterampilan untuk menganalisis dan menentukan metode pencarian yang efisien, yang akan berguna dalam merancang algoritma pemrograman di masa mendatang

Kegiatan pembelajaran ini akan mencakup,

- Tatap Muka (TM): 4 sesi, masing-masing 50 menit.
- Virtual Classroom (VC): 4 sesi, masing-masing 50 menit.
- Praktikum Terbimbing (PT): 4 sesi, masing-masing 60 menit.
- Bimbingan Mandiri (BM): 4 sesi, masing-masing 60 menit.

Sumber pembelajaran yang dapat diakses untuk mendukung pemahaman lebih lanjut tersedia di platform sekolah, dengan pustaka referensi yang telah disediakan.

Setelah mengikuti modul ini, mahasiswa diharapkan dapat mencapai tingkat pemahaman yang cukup, setidaknya 80%, sebelum melanjutkan ke materi berikutnya.

KEGIATAN BELAJAR 1

BINARY SEARCH & SEQUENTIAL SEARCH

A. Deskripsi Singkat

Materi ini dirancang untuk membantu mahasiswa memahami teknik-teknik pencarian dasar dalam algoritma, khususnya dalam mengenali dan membedakan ciri-ciri binary search dan sequential search. Pemahaman akan kedua teknik ini sangat penting, karena pencarian data adalah salah satu elemen inti dalam pengolahan data yang efisien dalam pemrograman. Dengan memahami kedua teknik ini, mahasiswa dapat menentukan metode pencarian yang tepat sesuai kebutuhan, serta mengoptimalkan kinerja pemrograman.

Pada kegiatan belajar ini, mahasiswa akan diperkenalkan pada cara kerja binary search dan sequential search, serta kapan sebaiknya setiap teknik digunakan. Dengan menguasai konsep ini, mahasiswa akan memiliki dasar logis yang kuat untuk merancang algoritma pencarian yang efisien dan sesuai konteks. Materi ini akan mencakup aturan dan langkah-langkah pseudocode untuk merancang kedua teknik pencarian ini, sehingga mahasiswa dapat memahami proses kerja pencarian data secara bertahap. Selain itu, mahasiswa akan belajar bagaimana binary search, dengan metode pencarian terurutnya, memungkinkan pencarian lebih cepat dibandingkan sequential search pada data besar yang terurut.

B. Relevansi

Modul ini merupakan modul lanjutan yang sangat penting setelah mahasiswa mempelajari dan mengimplementasikan konsep dasar algoritma dalam pemrograman, khususnya dalam penggunaan algoritma pencarian di bahasa C++. Pada modul ini, mahasiswa akan memperdalam pemahaman mereka terhadap dua teknik pencarian

utama, yaitu binary search dan sequential search, dengan fokus pada pengenalan ciri-ciri serta kelebihan dan kekurangan dari masing-masing teknik.

Sebagai modul lanjutan, materi ini dirancang untuk membantu mahasiswa tidak hanya memahami cara kerja kedua algoritma pencarian ini, tetapi juga menentukan situasi yang paling tepat untuk masing-masing metode berdasarkan kebutuhan dan jenis data yang diolah. Dengan menguasai binary search yang efisien pada data terurut dan sequential search yang sesuai untuk data tak terurut, mahasiswa akan memperoleh wawasan yang lebih dalam tentang optimalisasi algoritma dan akan mampu membandingkan berbagai pendekatan pencarian.

Dalam rangkaian modul ini, pemahaman tentang algoritma searching dengan menggunakan kedua teknik binary search dan sequential search sangat relevan terhadap pencapaian Sub-Capaian Pembelajaran Mata Kuliah (Sub-CPMK), karena memberikan dasar analitis bagi mahasiswa untuk dapat merancang algoritma secara logis dan efisien. Dengan pemahaman ini, akan membantu mahasiswa mengembangkan keterampilan dalam merancang algoritma yang akan diimplementasikan dalam berbagai bahasa pemrograman pada modul-modul selanjutnya.

C. Pembelajaran

Capaian pembelajaran yang diharapkan pada kegiatan belajar yang pertama ini adalah peserta kuliah akan mampu mengetahui dasar algoritma. Untuk mencapai kompetensi lulusan ini, peserta kuliah diharapkan membaca dan menyelesaikan Kegiatan Belajar 1 ini. Untuk mencapai capaian pembelajaran pada Kegiatan Belajar 1, peserta kuliah harus mengikuti urutan kegiatan pembelajaran yang telah disiapkan dalam modul ini.

1. Uraian

1.1 Pendahuluan

Algoritma search adalah serangkaian langkah atau instruksi yang digunakan untuk mencari elemen atau informasi tertentu di dalam suatu dataset. Tujuannya adalah untuk menemukan posisi atau keberadaan elemen yang dicari. Dalam pemrograman, algoritma search menjadi salah satu teknik penting dalam menyelesaikan berbagai masalah. Setelah proses pencarian dilaksanakan, akan diperoleh salah satu dari dua kemungkinan, yaitu data yang dicari ditemukan (successful) atau tidak ditemukan (unsuccessful). Ada dua macam teknik pencarian yaitu pencarian sekuensial (sequential search) dan pencarian biner (binary search). Perbedaan dari dua teknik ini terletak pada keadaan data. Pencarian sekuensial digunakan apabila data dalam keadaan acak atau tidak terurut. Sebaliknya, pencarian biner digunakan pada data yang sudah dalam keadaan urut.

1.2 Binary Search

Binary search adalah algoritma pencarian yang digunakan untuk menemukan posisi nilai target dalam array yang diurutkan. Algoritma ini bekerja dengan membagi interval pencarian menjadi dua secara berulang hingga nilai target ditemukan atau intervalnya kosong. Interval pencarian dibagi dua dengan membandingkan elemen target dengan nilai tengah ruang pencarian.

Contoh Kode Algoritma Binary Search

```
#include <iostream>
using namespace std;
```

```
// Fungsi Binary Search
```

```

int binarySearch(int arr[], int size, int
target) {

    int left = 0, right = size - 1;

    while (left <= right) {
        int mid = left + (right - left) / 2;

        // Cek apakah target berada di tengah
        if (arr[mid] == target)

            return mid;

        // Jika target lebih kecil, cari di
        bagian kiri
        if (arr[mid] > target)

            right = mid - 1;

        // Jika target lebih besar, cari di
        bagian kanan
        else

            left = mid + 1;
    }

    // Jika tidak ditemukan
    return -1;
}

```

Input	Output
Masukkan angka yang dicari: 7	Index: 3
Masukkan angka yang dicari: 4	Index: -1

Pseudocode Algoritma Searching

```
Procedure BinarySearch(Array A, Integer target)
    Integer left = 0
    Integer right = size of A - 1

    While left <= right
        Integer mid = (left + right) / 2

        If A[mid] == target
            Return mid // Target ditemukan,
            kembalikan indeks mid

        Else If A[mid] > target
            right = mid - 1 // Target lebih
            kecil, cari di bagian kiri

        Else
            left = mid + 1 // Target lebih
            besar, cari di bagian kanan

    EndWhile

    Return -1 // Target tidak ditemukan
EndProcedure
```

Penjelasan Proses:

1. Inisialisasi:

- left: variabel ini menunjukkan indeks pertama dari array (0).
- right: variabel ini menunjukkan indeks terakhir dari array (ukuran array - 1).

2. Proses Pencarian:

Selama nilai left kurang dari atau sama dengan right:

- Hitung mid sebagai indeks tengah, yaitu $(left + right) / 2$. Ini adalah titik tengah dari bagian array yang sedang dipertimbangkan.
- Bandingkan nilai $A[mid]$ dengan target:

Jika $A[mid] == \text{target}$: Target ditemukan di indeks mid, jadi algoritma mengembalikan nilai mid (indeks dari target).

Jika $A[mid] > \text{target}$: Target lebih kecil daripada nilai di mid, jadi kita akan mencari di sebelah kiri dari mid. Maka, kita ubah batas kanan right menjadi mid - 1.

Jika $A[mid] < \text{target}$: Target lebih besar daripada nilai di mid, jadi kita akan mencari di sebelah kanan dari mid. Maka, kita ubah batas kiri left menjadi mid + 1.

3. Menghentikan Pencarian:

Proses pencarian akan berhenti ketika left lebih besar daripada right, yang berarti target tidak ditemukan dalam array. Jika ini terjadi, algoritma mengembalikan nilai -1 yang menandakan bahwa target tidak ditemukan.

Misalkan kita memiliki array terurut:

$$A = [1, 3, 5, 7, 9, 11]$$

Dan target yang ingin dicari adalah 7.

1. Inisialisasi:

left = 0, right = 5 (indeks pertama dan terakhir).

2. Iterasi pertama:

mid = $(0 + 5) / 2 = 2$, sehingga $A[2] = 5$.

Karena $A[2] < 7$, kita mengubah left menjadi mid + 1 = 3.

3. Iterasi kedua:

left = 3, right = 5. Hitung mid: mid = $(3 + 5) / 2 = 4$, sehingga $A[4] = 9$.

Karena $A[4] > 7$, kita mengubah right menjadi mid - 1 = 3.

4. Iterasi ketiga:

left = 3, right = 3. Hitung mid: mid = $(3 + 3) / 2 = 3$, sehingga $A[3] = 7$.

Karena $A[3] == 7$, kita menemukan target pada indeks 3, dan algoritma mengembalikan 3.

1.3 Sequential Search

Salah satu ide yang muncul adalah dengan melakukan pengecekan kepada seluruh elemen yang ada. Dengan kata lain, kita akan melakukan:

- Periksa satu per satu dari sepatu pertama, kedua, ketiga, dan seterusnya.
- Jika ditemukan, langsung laporan.
- Jika sampai akhir belum juga ditemukan, artinya angka yang dicari tidak ada pada daftar.

Algoritma pencarian dengan membandingkan elemen satu per satu semacam ini disebut dengan sequential search atau linear

search . Jika ada N elemen pada daftar yang perlu dicari, maka paling banyak diperlukan N operasi perbandingan. Dalam kompleksitas waktu, performa algoritma sequential search bisa dinyatakan dalam $O(N)$

Contoh Kode Algoritma Sequential Searching

```
#include <iostream>

using namespace std;

int sequentialSearch(int arr[], int size, int x)
{
    for (int i = 0; i < size; i++) {
        if (arr[i] == x) {
            return i; // Nilai ditemukan,
            kembalikan indeksnya
        }
    }
    return -1; // Nilai tidak ditemukan,
    kembalikan -1
}

int main() {
    int arr[5] = {2, 3, 4, 10, 40};
    int x = 3;
    int n = 5;

    int result = sequentialSearch(arr, n, x);
    if (result == -1)
```

```
        cout << "Nilai tidak ditemukan" << endl;
    else
        cout << "Nilai ditemukan pada indeks: "
        << result << endl;

    return 0;
}
```

Output : Nilai ditemukan pada indeks: 1

Pseudocode Algoritma Sequential Searching

```
Procedure SequentialSearch(Array A, Integer
target)
    For i = 0 to size of A - 1
        If A[i] == target
            Return i // Jika nilai ditemukan,
            kembalikan indeks i
        EndIf
    EndFor
    Return -1 // Jika nilai tidak ditemukan,
    kembalikan -1
EndProcedure
```

- For loop: Mulai dengan indeks $i = 0$ hingga indeks terakhir dalam array ($\text{size} - 1$).
- If statement: Di setiap iterasi, bandingkan elemen array pada indeks i dengan nilai target.

- Jika nilai yang ada di $A[i]$ sama dengan target, maka kita kembalikan indeks i.
- Jika tidak ada elemen yang sama dengan target setelah seluruh array diperiksa, kembalikan -1 untuk menunjukkan bahwa nilai target tidak ditemukan dalam array.

1. Input:

Array: {2, 3, 4, 10, 40}

Target: 3

2. Proses:

Iterasi pertama: $arr[0] = 2$ tidak cocok dengan $x = 3$.

Iterasi kedua: $arr[1] = 3$ cocok dengan $x = 3$. Maka, indeks 1 dikembalikan.

3. Output:

Nilai ditemukan pada indeks: 1

2. Rangkuman

Secara umum, kedua algoritma pencarian memiliki karakteristik yang berbeda. Sequential search tidak memerlukan data yang terurut dan memiliki kompleksitas waktu $O(N)$, di mana N adalah ukuran data. Algoritma ini lebih baik diterapkan jika pencarian hanya dilakukan sekali. Sebaliknya, binary search memerlukan data yang terurut dan memiliki kompleksitas waktu $O(\log N)$, yang lebih efisien untuk ukuran data yang besar. Algoritma ini sangat cocok digunakan jika pencarian perlu dilakukan berkali-kali karena dapat mempercepat proses pencarian dengan membagi data menjadi dua bagian pada setiap langkah..

3. Pustaka.

[1] Aji, Alham Fikri dan Gozali, William. Pemrograman Kompetitif Dasar. ISBN 978-602-6598-89-9.

[2] Fariza, Arna. Setiowati, Yuliana. Algoritma Pencarian (Searching Algorithm).

D. Penilaian Kegiatan Belajar

1. Penilaian

1.1 Latihan

- 1) Misalkan Anda diberikan sebuah buku telepon yang berisikan beberapa kerabat anda. Anda sedang ingin menelpon beberapa orang dan anda sedang terburu-buru untuk segera menelponnya. Ternyata, Anda tidak perlu mencari nama satu per satu, halaman demi halaman. Anda dapat mencari nama tersebut cukup dalam beberapa perbandingan. Buku telepon memiliki sifat khusus, yaitu terurut berdasarkan abjad. Dengan sifat ini, Anda bisa membuka halaman tengah dari buku telepon, lalu periksa apakah nama yang Anda cari ada pada halaman tersebut. Jika nama yang Anda cari berada di sebelah kiri halaman tengah, Anda akan melanjutkan pencarian hanya di bagian kiri, dan sebaliknya jika berada di sebelah kanan. Dengan cara ini, setiap perbandingan akan mengeliminasi separuh rentang pencarian.

Buatlah program dalam C++ untuk mengimplementasikan algoritma Binary Search yang dapat mencari nama dalam daftar nama orang pada buku telepon yang terurut berdasarkan abjad. Program harus menampilkan apakah



nama tersebut ditemukan atau tidak, beserta indeksnya jika ditemukan.

Daftar Nama:

```
string names[] = {  
    "Ahmad", "Alif", "Bella", "Budi", "Chandra", "Citra", "Diana",  
    "Dewi", "Eli", "Eko", "Fajar", "Farhan", "Gani", "Gita", "Hani",  
    "Hendra", "Indra", "Ika", "Jasmine", "Joko", "Kirana", "Kevin",  
    "Lara", "Laras", "Lina", "Marcel", "Maya", "Nadia", "Nina",  
    "Oki", "Omar", "Putri", "Qiana", "Rina", "Rudi", "Sari", "Sinta",  
    "Tari", "Tina", "Uli", "Umar", "Vina", "Vira", "Wanda", "Wira",  
    "Xena", "Yani", "Yudi", "Zaki"};
```

Format Masukan dan Keluaran:

Baris pertama merupakan n test case

Baris kedua hingga ke baris-n merupakan nama yang dicari



Keluaran terdiri dari n-baris yang merupakan indeks dari nama yang dicari (keluarkan “tidak ditemukan” jika nama yang dicari tidak terdapat pada data buku telepon yang telah diberikan)

Format masukan :

Input	Output
Marcel	25
Hani	14
joko	tidak ditemukan

1.2 Tantangan



1) Binary Search

<https://tlx.toki.id/courses/competitive-1/chapters/03/problems/E>

Deskripsi

Pak Dengklek memiliki N ekor bebek. Bebek ke-i memiliki berat A_i . Pak Dengklek juga memiliki Q buah pertanyaan. Pertanyaan ke-ii berbunyi: berapa banyak bebek yang memiliki berat lebih dari ($>$) x_i dan kurang dari sama dengan (\leq) y_i ? Jawablah pertanyaan-pertanyaan tersebut!

Masukan

Masukan diberikan dalam format berikut:

N

A₁ A₂ ... A_N

Q

x₁ y₁

x₂ y₂

:

x_Q y_Q

Format Keluaran

Untuk setiap pertanyaan, keluarkan sebuah baris berisi banyaknya bebek yang dimaksud.

Format masukan :

Input	Output
5	4
3 6 8 10 20	1

2	
2 15	
10 20	

2) Sequential Search

<https://tlx.toki.id/courses/competitive-1/chapters/03/problems/A>

Deskripsi

Pak Dengklek memiliki data yang terdiri atas N buah bilangan bulat: A₁ hingga A_N. Ia juga memiliki sebuah bilangan bulat X. Ia ingin tahu, di antara data tersebut, bilangan mana yang selisihnya dengan X paling kecil?

Masukan

Masukan diberikan dalam format berikut:

N X

A₁ A₂ ... A_N

Keluaran

Keluarkan sebuah baris berisi sebuah bilangan bulat dari data Pak Dengklek yang memiliki selisih terkecil dengan X.

Format masukan :

Input	Output
3 101234 101231 101237 100000	4 101231
5 12345 100005 -100001 3 10004 -2	10004

2. Umpan Balik dan Tindak Lanjut

1) Binary Search

Program Binary Search

```
#include <iostream>
#include <string>
using namespace std;

// Fungsi untuk melakukan binary search pada
array nama

int binarySearch(string names[], string target,
int size)

{
    int left = 0;
    int right = size - 1;

    while (left <= right)
    {
        int mid = left + (right - left) / 2;

        if (names[mid] == target)
        {
            return mid; // Kembalikan indeks
            jika ditemukan
        }
        else if (names[mid] < target)
        {
            left = mid + 1; // Abaikan bagian
            kiri
```

```
        }

        else

        {

            right = mid - 1; // Abaikan bagian
            kanan

        }

    return -1; // Tidak ditemukan
}

int main()
{
    int N;
    cin >> N;
    string target;

    // Menyimpan hasil pencarian
    string results[N];

    // Daftar nama untuk pencarian
    string names[] = {
        "Ahmad", "Alif", "Bella", "Budi",
        "Chandra", "Citra",
        "Diana", "Dewi", "Eli", "Eko", "Fajar",
        "Farhan",
        "Gani", "Gita", "Hani", "Hendra",
        "Indra", "Ika",
```

```
        "Jasmine", "Joko", "Kirana", "Kevin",
        "Lara", "Laras",
        "Lina", "Marcel", "Maya", "Nadia",
        "Nina", "Oki",
        "Omar", "Putri", "Qiana", "Rina",
        "Rudi", "Sari",
        "Sinta", "Tari", "Tina", "Uli", "Umar",
        "Vina",
        "Vira", "Wanda", "Wira", "Xena", "Yani",
        "Yudi", "Zaki"
    };

    int size = sizeof(names) / sizeof(names[0]);

    // Input nama yang akan dicari
    for (int i = 0; i < N; i++)
    {
        cin >> target;

        int result = binarySearch(names, target,
size);

        // Menyimpan hasil pencarian ke array
results
        if (result != -1)
        {
            results[i] = to_string(result);
        }
        else
```

```
{  
    results[i] = "tidak ditemukan";  
}  
  
}  
  
// Mencetak semua hasil pencarian setelah  
semua input selesai  
  
for (int i = 0; i < N; i++)  
{  
    cout << results[i] << endl;  
}  
  
return 0;  
}
```



ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA

MODUL VIII

*Sorting &
Kompleksitas
Algoritma*



Elly Warni
Ingrid Nurtanio
Amil Ahmad Ilham
Adnan
Anugrayani Bustamin

PRAKATA

Alhamdulillah, puji dan syukur kami panjatkan ke hadirat Allah SWT yang telah melimpahkan rahmat dan hidayah-Nya, sehingga modul ini dapat terselesaikan dengan baik. Shalawat serta salam senantiasa tercurah kepada junjungan kita, Nabi Muhammad SAW, beserta keluarga dan sahabat beliau yang telah membawa kita dari zaman kegelapan menuju terang benderang ilmu pengetahuan.

Modul ini disusun sebagai bagian dari rangkaian pembelajaran yang bertujuan untuk membantu mahasiswa memahami algoritma sorting dan kompleksitas algoritma. Modul ini merupakan kelanjutan dari modul sebelumnya yang membahas dasar algoritma dan beberapa fungsi dalam algoritma pemrograman, di mana pada modul ini mahasiswa mulai menerapkan konsep-konsep tersebut dalam pemahaman serta implementasi algoritma sorting dalam bahasa pemrograman, serta melakukan analisis terhadap kompleksitas algoritma.

Tidak dapat dipungkiri bahwa proses penyusunan modul ini tidak lepas dari dukungan berbagai pihak. Oleh karena itu, kami mengucapkan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan masukan dan bimbingan dalam proses penyusunan modul ini. Terima kasih kami sampaikan kepada rekan-rekan dosen dan staf yang telah memberikan kritik dan saran yang membangun, sehingga modul ini dapat terselesaikan dengan baik.

Kami menyadari bahwa modul ini masih memiliki keterbatasan dan kekurangan. Oleh karena itu, kami sangat mengharapkan masukan, kritik, dan saran dari para pembaca untuk penyempurnaan di masa yang akan datang. Semoga modul ini dapat memberikan manfaat dan menjadi bahan referensi yang berharga bagi para mahasiswa dalam memahami konsep algoritma dan pemrograman.



Akhir kata, semoga upaya ini mendapat ridha Allah SWT dan memberikan kontribusi positif bagi kemajuan ilmu pengetahuan. Amin.

Makassar, November 2024

Penulis

DAFTAR ISI

.....	1
PRAKATA	i
DAFTAR ISI	iii
RPS MATA KULIAH ALGORITMA DAN STRUKTUR DATA	1
MODUL ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA	12
Pengantar.....	12
KEGIATAN BELAJAR 1.....	14
A. Deskripsi Singkat	14
B. Relevansi	14
C. Pembelajaran	15
D. Penilaian Kegiatan Belajar	24
KEGIATAN BELAJAR 2.....	29
A. Deskripsi Singkat	29
B. Relevansi	29
C. Pembelajaran	30
D. Penilaian Kegiatan Belajar	42
KEGIATAN BELAJAR 3.....	50
A. Deskripsi Singkat	50
B. Relevansi	50
C. Pembelajaran	51
D. Penilaian Kegiatan Belajar	59

RPS MATA KULIAH ALGORITMA DAN STRUKTUR DATA

	UNIVERSITAS HASANUDDIN FAKULTAS TEKNIK PROGRAM STUDI TEKNIK INFORMATIKA					Kode Dokumen
RENCANA PEMBELAJARAN SEMESTER						
MATA KULIAH (MK)	KODE	Rumpun MK	BOBOT (skt)		SEMESTER	Tgl Penyusunan
Algoritma Pemrograman dan Struktur Data	23D12110604	Teknik Informatika	T=2	P=2	2	19-02-2023
OTORISASI	Pengembang RPS	Koordinator RMK		Ketua PRODI		
	Dr. Ir. Ingrid Nurtanio., MT Dr. Amil Ahmad Ilham., ST., M.IT Adnan., ST., MT., PhD Anugrayani Bustamin., ST., MT Elly Warni, S.T., M.T	Dr. Ir. Ingrid Nurtanio., MT		Prof. Dr. Indrabayu., ST., M.T., M.Bus.Sys		
	CPL-PRODI yang dibebankan pada MK <i>Intended Learning Outcomes</i>					

Capaian Pembelajaran (CP)	CPL 1	Memiliki dasar pengetahuan Teknik Informatika yang meliputi teori dan konsep dasar dari Ilmu Komputer, Matematika dan Statistika, Algoritma dan Pemrograman, Rekayasa Perangkat Lunak, Manajemen Informasi dan Ketahanan Digital, serta pengetahuan tingkat lanjut pada bidang-bidang khusus Teknik Informatika, seperti Kecerdasan Buatan, Data Science, Jaringan Komputer, Komputasi Awan dan Internet of Things.
	CPL 3	Mampu mengaplikasikan pengetahuan bidang Teknik Informatika yang dipadankan dengan bidang ilmu lainnya untuk menganalisa dan mencari solusi dari berbagai masalah berbasis komputasi.
	CPL 4	Mampu mendesain, mengimplementasikan dan mengevaluasi solusi berbasis komputasi dengan mengaplikasikan ilmu Teknik Informatika dan dasar-dasar pembangunan perangkat lunak.
	Capaian Pembelajaran Mata Kuliah (CPMK)	
	CPMK	Setelah mengikuti Mata Kuliah Algoritma Pemrograman dan Struktur Data selama 1 (satu) semester, mahasiswa mampu menguasai dan menerapkan konsep teoritis bidang Informatika dan ilmu komputer secara dan mendalam serta memformulasikan penyelesaian masalah procedural dengan pemikiran logis dan sistematis secara mandiri dan terukur sebagai dasar pengembangan perangkat lunak.

	CPL ⇒ Sub-CPMK	
	CPL 1	Sub-CPMK 1 : Mahasiswa mampu mendefinisikan konsep dan pengertian Algoritma
	CPL 1	Sub-CPMK 2 : Mahasiswa mampu mendeksripsikan Dasar Algoritma
	CPL 3	Sub-CPMK 3 : Mahasiswa mampu mengimplementasikan Flowchart
	CPL 1	Sub-CPMK 4 : Mahasiswa mampu memahami bentuk pemilihan dan perulangan
	CPL 4	Sub-CPMK 5 : Mahasiswa mampu mengimplementasikan prosedur, fungsi dan pemrosesan teks
	CPL 1	Sub-CPMK 6 : Mahasiswa mampu mempraktekkan konsep array
	CPL 4	Sub-CPMK 7 : Mahasiswa mampu menetapkan ciri dari teknik pencarian
	CPL 4	Sub-CPMK 8 : Mahasiswa mampu menetapkan ciri dari teknik sorting dan menganalisis complexitas algoritma
	CPL 1	Sub-CPMK 9 : Mahasiswa mampu memahami struct dan konsep linked list
	CPL 1	Sub-CPMK 10 : Mahasiswa mampu memahami Antrian dan Stack
	CPL 1	Sub-CPMK 11 : Mahasiswa mampu memahami teknik graph
	CPL 3	Sub-CPMK 12 : Mahasiswa mampu mengimplementasi konsep Tree
Deskripsi Singkat MK	<p>Mata kuliah Algoritma Pemrograman dan Struktur Data merupakan mata kuliah yang menyajikan konsep pemrograman secara mendalam terkait algoritma dan struktur data. Muatan mata kuliah merupakan dasar dari ilmu matematika, algoritma dan logika yang menjadi syarat utama pada mata kuliah Pemrograman lanjut . Mata kuliah ini disajikan untuk mahasiswa semester dua di Program Studi Teknik Informatika, Penanggungjawab mata kuliah adalah Laboratorium Komputer . Metode Pembelajaran bauran, teori dan praktik</p>	
Bahan Kajian / Materi Pembelajaran	<ol style="list-style-type: none"> 1. Konsep Algoritma 2. Dasar Algoritma (Aturan Penulisan Header, deklarasi dan deskripsi) 3. Flowchart (Simbol, variabel, algoritma cabang, runtunan) 4. Pemilihan (Alir Kontrol dan Pengulangan) 5. Prosedur, Fungsi dan Pemrosesan Teks 6. Array – Larik 7. Pencarian (Binary Search) 	

	<ul style="list-style-type: none"> 8. Pencarian (Sequential Search) 9. Bubble dan Insertion Sort 10. Merge dan selection sort 11. Quick sort dan complexita algoritma 12. Struct dan Linked List 13. Antrian dan Stack 14. Graph berarah dan tidak berarah 15. Tree
--	---

Pustaka	Utama :	
	<ul style="list-style-type: none"> 1. Munir, Rinaldi dan Lidya, Leony. 2016. Algoritma dan Pemrograman dalam Bahasa Pascal, C dan C++ Edisi Keenam. Informatika : Bandung 2. Gozali, William dan Aji, Alham Fikri. 2014. Pemrograman Kompetitif Dasar. Ikatan Alumni Tim Olimpiade Komputer Indonesia. 3. Zakaria, Teddy M dan Prijono, Agus. 2005. Konsep dan Implementasii Struktur Data. Informatika : Bandung. 4. Niswar, M., Ilham, A., Zainuddin, Z., Adnan, A., P, A., Warni, E., Aswad, I., & Muslimin, Z. (2021). Sosialisasi Metode Berfikir Komputasional pada Pendidikan Dasar dan Menengah di Lingkup Sulawesi Selatan. URNAL EPAT eknologi erapan ntuk engabdian asyarakat, 4(1), 46-52. https://doi.org/10.25042/jurnal_tepat.v4i1.172 	
	Pendukung:	
	Tuliskan pustaka pendukung jika ada, sebagai pengayaan literasi	

Dosen Pengampu		1. Dr. Ir. Ingrid Nurtanio., MT 2. Dr. Amil Ahmad Ilham., ST., M.IT 3. Adnan., ST., MT., PhD 4. Anugrayani Bustamin., ST., MT 5. Elly Warni, S.T., M.T						
Matakuliah syarat		Dasar Pemrograman Komputer						
Pekan Ke-	Sub-CPMK (Kemampuan akhir tiap tahapan belajar)	Penilaian		Metode Pembelajaran, Penugasan Mahasiswa, [Estimasi Waktu]			Materi Pembelajaran [Pustaka]	Bobot Penilaian (%)
		Indikator	Kriteria & Bentuk	Luring (offline)	Daring (online)			
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	
1	Mahasiswa mampu mengetahui konsep dan pengertian Algoritma	Mengetahui konsep definisi, manfaat, mekanisme dan kontribusi Algoritma	Assignments	Lecture TM: 4 x 50 menit.	Lecture VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Konsep dan definisi algoritma 2. Mekanisme dan Kontribusi Algoritma	5%	

2	Mahasiswa mampu memahami Dasar Algoritma CPL 3- SubCPMK 2	Memahami Dasar Algoritma (tipe data, aturan penulisan, proses, instruksi dan aksi)	quizzes	Lecture, case study TM: 4 x 50 menit.	Lecture, case study VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Dasar Algoritma (tipe data, aturan penulisan, proses, instruksi, dan aksi) Pustaka : [1]	5%
3	Mahasiswa mampu mengenali Flowchart CPL 4- SubCPMK 3	Mampu mengenali Flowchart (simbol, variabel, algoritma cabang, dan runtunan)	Assignments	Lecture, small group discussion TM: 4 x 50 menit.	Lecture, small group discussion VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit	Flowchart (Simbol, variabel, algoritma cabang, runtunan) Pustaka : [1]	7%
4	Mahasiswa mampu memahami bentuk pemilihan (alih kontrol dan pengulangan)	Mampu memahami bentuk pemilihan (alih kontrol dan pengulangan)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses	Pemilihan Kontrol (Alir dan Pengulangan) Pustaka : [1]	7%

					/104D4224/inde x.php?id_session =13610		
5	Mahasiswa mampu mengimplementasikan prosedur, fungsi dan pemrosesan teks	Mampu mengimplementasikan pemanggilan prosedur, pemanggilan fungsi dan pemrosesan teks	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Pemanggilan Prosedur 2. Pemanggilan Fungsi 3. Pemrosesan Teks Pustaka : [1]	7%
6	Mahasiswa mampu mempraktekkan konsep array	Mampu mempraktekkan konsep array seperti menginisialisasi array, mengisi elemen, mencari nilai maksimum dan minimum	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Menginisialisasi array 2. Mengisi elemen array 3. Mencari nilai maksimum dan minimum Pustaka : [1][2]	7%

7	Mahasiswa mampu menetapkan ciri dari teknik pencarian	Mampu menetapkan ciri dari teknik pencarian ● Binary Search ● Sequential Search	Assignments quizzes	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Binary Search 2. Sequential Search Pustaka : [2][3]	7%
8	Evaluasi Tengah Semester / Ujian Tengah Semester						
9	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting ● Bubble Sort ● Insertion Sort	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Bubble Sort 2. Insertion Sort Pustaka : [2][3]	9%
	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting	Assignments	Lecture, practicum	Lecture, practicum VC: 4 x 50 menit.	1. Merge Sort Selection Sort Pustaka : [2][3]	9%

10		Merge Sort Selection Sort		TM: 4 x 50 menit.	PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610		
11	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting Quick Sort berserta complexitas Algoritmanya	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Quick Sort 2. Complexitas Algoritma Pustaka : [2][3]	9%
12	Mahasiswa mampu memahami struct dan konsep linked list	Mampu memahami struct dan konsep dan operasi dasar linked list (deklarasi simpul, penambahan)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit	1. Konsep Struct dan implementasinya 2. Deklarasi list 3. Penambahan simpul pada list awal, akhir dan tengah	7%

		(simpul pada linked list awal, akhir dan tengah)			https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Pustaka : [2][3]	
13	Mahasiswa mampu memahami Antrian dan Stack	Mampu memahami Antrian dan Stack dalam List	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Antrian dan Stack dalam list Pustaka : [2][3]	7%
14	Mahasiswa mampu memahami teknik graph	mampu memahami teknik graph (graph berarah dan tidak berarah menggunakan linked list)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Graph berarah Graph tidak berarah Pustaka : [2][3]	7%

					4224/index.php? id_session=1361 0		
15	Mahasiswa mampu mengimplementasi konsep Tree	Mampu mengimplementasi konsep Tree menggunakan linked list	Assignments quizzes	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Konsep tree menggunakan linked List Pustaka : [2][3]	7%
16	Evaluasi Akhir Semester / Ujian Akhir Semester						

MODUL ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA

MODUL VIII

SORTING & KOMPLEKSITAS ALGORITMA



Pengantar

Modul ini dirancang untuk membantu mahasiswa memahami dan mengenali ciri-ciri dari berbagai teknik pengurutan dalam algoritma, khususnya algoritma sorting seperti Bubble Sort, Insertion Sort, Merge Sort, Selection Sort, dan Quick Sort, serta analisis kompleksitas algoritma. Modul ini merupakan langkah penting untuk mencapai capaian pembelajaran mata kuliah (CPMK) yang telah ditetapkan, di mana mahasiswa diharapkan dapat memahami dan membedakan berbagai metode pengurutan serta menganalisis kompleksitas waktu dan ruang dari algoritma yang digunakan.

Pada Sub-CPMK ini, mahasiswa diharapkan mampu memahami konsep dan karakteristik dari berbagai algoritma sorting sebagai teknik dasar dalam mengurutkan data. Dengan mempelajari beberapa jenis algoritma sorting, mahasiswa akan memahami bagaimana pengurutan dilakukan secara berulang untuk menyusun data ke dalam urutan yang benar. Selain itu, mahasiswa juga akan diajarkan untuk menganalisis kompleksitas waktu dari algoritma-algoritma tersebut, serta memahami bagaimana faktor ukuran data dapat mempengaruhi efisiensi algoritma.

Kegiatan pembelajaran dalam modul ini akan dilakukan melalui diskusi kelompok kecil dan pembelajaran interaktif, didukung oleh platform pembelajaran daring. Mahasiswa akan mempelajari cara kerja beberapa algoritma sorting, baik secara teoritis maupun dalam penerapan pada

koe pemrograman Dengan pemahaman ini, mahasiswa akan memiliki keterampilan untuk menganalisis dan menentukan metode sorting yang efisien berdasarkan jenis data dan konteks penggunaannya, yang sangat berguna dalam merancang algoritma pemrograman di masa mendatang.

Kegiatan pembelajaran ini akan mencakup,

- Tatap Muka (TM): 4 sesi, masing-masing 50 menit.
- Virtual Classroom (VC): 4 sesi, masing-masing 50 menit.
- Praktikum Terbimbing (PT): 4 sesi, masing-masing 60 menit.
- Bimbingan Mandiri (BM): 4 sesi, masing-masing 60 menit.

Sumber pembelajaran yang dapat diakses untuk mendukung pemahaman lebih lanjut tersedia di platform sekolah, dengan pustaka referensi yang telah disediakan.

Setelah mengikuti modul ini, mahasiswa diharapkan dapat mencapai tingkat pemahaman yang cukup, setidaknya 80%, sebelum melanjutkan ke materi berikutnya.

KEGIATAN BELAJAR 1

BUBBLE SORT & INSERTION SORT

A. Deskripsi Singkat

Materi ini dirancang untuk membantu mahasiswa memahami teknik-teknik dasar dalam algoritma pengurutan, khususnya dalam mengenali dan membedakan ciri-ciri Bubble Sort dan Insertion Sort. Pemahaman tentang kedua algoritma ini sangat penting, karena pengurutan data adalah salah satu elemen utama dalam pemrograman untuk menyusun data dengan urutan yang benar. Dengan memahami kedua algoritma ini, mahasiswa dapat menentukan metode pengurutan yang sesuai dengan kebutuhan dan mengoptimalkan kinerja program.

Pada kegiatan belajar ini, mahasiswa akan diperkenalkan pada cara kerja Bubble Sort dan Insertion Sort, serta kapan sebaiknya setiap teknik digunakan. Dengan menguasai konsep ini, mahasiswa akan memiliki dasar yang kuat untuk merancang algoritma pengurutan yang efisien dan sesuai dengan konteksnya. Materi ini akan mencakup aturan dan langkah-langkah pseudocode untuk merancang kedua algoritma pengurutan ini, sehingga mahasiswa dapat memahami proses kerja pengurutan data secara bertahap. Selain itu, mahasiswa akan belajar bagaimana Bubble Sort, dengan metode perbandingan berulang, mengurutkan data dengan memindahkan elemen terbesar ke posisi akhir, sementara Insertion Sort, dengan menyisipkan elemen ke posisi yang tepat, dapat lebih efisien pada data yang hampir terurut.

B. Relevansi

Modul ini mempelajari dua algoritma pengurutan dasar, yaitu Bubble Sort dan Insertion Sort. Kedua algoritma ini sangat penting untuk dipahami karena mereka adalah teknik dasar dalam pengurutan data yang banyak digunakan dalam pemrograman. Pada modul ini, mahasiswa akan

mempelajari cara kerja kedua algoritma ini, termasuk langkah-langkah yang terlibat dalam proses pengurutan data.

Sebagai modul lanjutan, materi ini dirancang untuk membantu mahasiswa memahami konsep awal algoritma sorting dan cara kerja kedua algoritma sorting ini. Dengan menguasai kegiatan belajar 1 ini, mahasiswa akan memperoleh wawasan mengenai algoritma sorting dan akan sangat diperlukan untuk memahami berbagai jenis sorting yang nantinya akan dipelajari lebih banyak lagi pada kegiatan belajar berikutnya.

Modul ini juga memberikan pengenalan tentang cara menganalisis kompleksitas algoritma untuk menilai efisiensi masing-masing algoritma dalam hal waktu dan ruang. Pemahaman yang diperoleh dari modul ini akan sangat berguna sebagai dasar untuk modul-modul sorting selanjutnya, di mana mahasiswa akan mempelajari berbagai algoritma pengurutan lainnya yang lebih kompleks dan efisien.

C. Pembelajaran

Capaian pembelajaran yang diharapkan pada kegiatan belajar yang pertama ini adalah peserta kuliah akan mampu mengetahui dasar algoritma. Untuk mencapai kompetensi lulusan ini, peserta kuliah diharapkan membaca dan menyelesaikan Kegiatan Belajar 1 ini. Untuk mencapai capaian pembelajaran pada Kegiatan Belajar 1, peserta kuliah harus mengikuti urutan kegiatan pembelajaran yang telah disiapkan dalam modul ini.

1. Uraian

1.1 Pendahuluan

Sorting adalah proses pengurutan data baik dari nilai tertinggi maupun dari nilai terendah. Ada banyak algoritma popular untuk

mengurutkan data seperti Bubble Sort, Selection Sort, Quick Sort, Insertion Sort, Merge Sort.

Algoritma-algoritma pengurutan ini biasanya dibedakan berdasarkan:

- Kompleksitas perbandingan antar elemen (terkait dengan kasus terbaik dan terburuk) dinotasikan dengan $O(n \log n)$ untuk pencarian yang baik, dan $O(n^2)$ sebagai kasus yang buruk.
- Kompleksitas pertukaran elemen, terkait dengan cara yang digunakan elemen setelah dibandingkan.
- Penggunaan memori. Ada beberapa jenis algoritma yang memerlukan memori sementara untuk menyimpan list
- Rekursif.
- Metode-metode penggunaanya, seperti exchange, insertion, partition, merging, dan selection.

1.2 Bubble Sort

4	3	2	8	5	3
3	4	2	8	5	3
3	2	4	8	5	3
3	2	4	8	5	3
3	2	4	5	8	3
3	2	4	5	3	8

■ ditukar
■ tidak ditukar

Bubble Sort adalah metode pengurutan algoritma dengan cara melakukan penukaran data secara terus menerus sampai bisa dipastikan dalam suatu iterasi tertentu tidak ada lagi perubahan/penukaran. Algoritma ini menggunakan perbandingan dalam operasi antar elemennya.

Bubble sort memproses setiap pasang elemen yang bersebelahan satu per satu. Mulai dari elemen pertama, cek apakah elemen sesudahnya (yaitu elemen kedua) lebih kecil. Bila ya, artinya elemen pertama ini harus terletak sesudah elemen kedua. Untuk itu, lakukan penukaran. Bila tidak, tidak perlu lakukan penukaran. Lanjut periksa elemen kedua, ketiga, dan seterusnya. Proses ini mengakibatkan elemen dengan nilai terbesar pasti digiring ke posisi terakhir.

Proses pengurutan Bubble Sort dapat dijelaskan dalam beberapa langkah berikut:

1. Iterasi melalui Array: Mulai dari elemen pertama, bandingkan setiap pasangan elemen yang berdekatan.
2. Tukar Elemen: Jika elemen pertama lebih besar daripada elemen kedua, tukar posisi mereka. Ini memastikan bahwa elemen yang lebih besar “menggelembung” ke akhir array setelah setiap iterasi.
3. Lanjutkan ke Elemen Berikutnya: Pindah ke pasangan elemen berikutnya dan ulangi proses perbandingan dan penukaran hingga akhir array.
4. Pengulangan Proses: Setelah menyelesaikan satu iterasi melalui array, ulangi proses dari awal. Setiap kali, elemen terbesar akan berada pada posisinya yang benar. Oleh karena itu, pada setiap iterasi berikutnya, kita dapat mengecualikan elemen terakhir dari perbandingan karena sudah berada di tempat yang benar.
5. Penghentian: Proses ini berlanjut sampai tidak ada lagi pertukaran yang diperlukan, yang berarti array sudah terurut.

Contoh Kode Algoritma Bubble Sort

```
#include <iostream>
using namespace std;

void bubbleSort(int arr[], int n) {
    for (int i = 0; i < n - 1; i++)
        for (int j = 0; j < n - i - 1; j++)
            if (arr[j] > arr[j + 1])
                swap(arr[j], arr[j + 1]);
}

int main() {
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n = sizeof(arr) / sizeof(arr[0]);

    bubbleSort(arr, n);

    for (int i : arr) cout << i << " ";
    cout << endl;

    return 0;
}
```

Output : 11 12 22 25 34 64 90

Pseudocode Algoritma Bubble Sort

```
function bubbleSort(array, n)
    for i from 0 to n-2
```

```
for j from 0 to n-2
    if array[j] > array[j + 1]
        swap(array[j], array[j + 1])
```

Penjelasan Proses:

1. Inisialisasi:

- left: variabel ini menunjukkan indeks pertama dari array (0).
- right: variabel ini menunjukkan indeks terakhir dari array (ukuran array - 1).

2. Proses Pencarian:

Selama nilai left kurang dari atau sama dengan right:

- Hitung mid sebagai indeks tengah, yaitu $(left + right) / 2$. Ini adalah titik tengah dari bagian array yang sedang dipertimbangkan.
- Bandingkan nilai $A[mid]$ dengan target:

Jika $A[mid] == \text{target}$: Target ditemukan di indeks mid, jadi algoritma mengembalikan nilai mid (indeks dari target).

Jika $A[mid] > \text{target}$: Target lebih kecil daripada nilai di mid, jadi kita akan mencari di sebelah kiri dari mid. Maka, kita ubah batas kanan right menjadi mid - 1.

Jika $A[mid] < \text{target}$: Target lebih besar daripada nilai di mid, jadi kita akan mencari di sebelah kanan dari mid. Maka, kita ubah batas kiri left menjadi mid + 1.

3. Menghentikan Pencarian:

Proses pencarian akan berhenti ketika left lebih besar daripada right, yang berarti target tidak ditemukan dalam array. Jika ini

terjadi, algoritma mengembalikan nilai -1 yang menandakan bahwa target tidak ditemukan.

Misalkan kita memiliki array terurut:

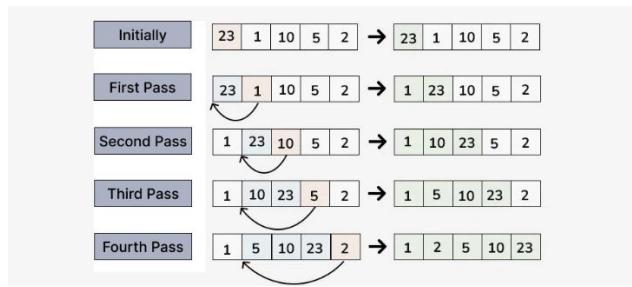
A = [1, 3, 5, 7, 9, 11]

Dan target yang ingin dicari adalah 7.

1. Inisialisasi: fungsi bubbleSort menerima array dan integer n yang merupakan jumlah elemen dalam array Iterasi pertama:
2. Loop Pertama (for i from 0 to n-2): Loop pertama mengontrol jumlah iterasi besar yang dilakukan oleh algoritma. Setiap iterasi ini mengurangi jumlah elemen yang harus dibandingkan, karena elemen terbesar "mengambang" ke posisi yang benar di akhir array.
3. Loop Kedua (for j from 0 to n-i-2): Loop kedua mengontrol perbandingan antar elemen dalam array. Dalam setiap iterasi, elemen array[j] dan array[j + 1] dibandingkan.
4. Perbandingan dan Pertukaran: Jika elemen saat ini (array[j]) lebih besar daripada elemen berikutnya (array[j + 1]), maka keduanya ditukar sehingga elemen yang lebih besar bergerak ke kanan.

Output Akhir: Setelah kedua loop selesai, array akan terurut dalam urutan menaik..

1.3 Insertion Sort



Insertion Sort adalah algoritma pengurutan sederhana yang bekerja dengan menyiapkan secara berulang-ulang setiap elemen dari daftar yang belum diurutkan ke dalam posisi yang benar di bagian daftar yang telah diurutkan. Kita mulai dengan elemen kedua dari array karena elemen pertama dalam array diasumsikan sudah terurut. Kemudian bandingkan elemen kedua dengan elemen pertama dan periksa apakah elemen kedua lebih kecil, lalu tukar. Pindah ke elemen ketiga dan bandingkan dengan dua elemen pertama dan letakkan pada posisi yang benar. Lalu Ulangi sampai seluruh array terurut.

Contoh Kode Algoritma Insertion Sort

```
#include <iostream>
using namespace std;

void insertionSort(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;

        // Pindahkan elemen arr[0..i-1] yang
        // lebih besar dari key ke satu posisi ke depan
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
}
```

```
        j--;
    }

    arr[j + 1] = key; // Tempatkan key di
posisi yang tepat
}

}

int main() {
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n = sizeof(arr) / sizeof(arr[0]);

    insertionSort(arr, n);

    // Cetak array yang telah diurutkan
    for (int i : arr) {
        cout << i << " ";
    }
    cout << endl;

    return 0;
}
```

Output : 11 12 22 25 34 64 90

Pseudocode Algoritma Insertion Sort

```
function insertionSort(array, n)
```

```
for i from 1 to n-1
    key = array[i]
    j = i - 1

    while j >= 0 and array[j] > key
        array[j + 1] = array[j]
        j = j - 1

    array[j + 1] = key
```

- Inisialisasi: Fungsi insertionSort menerima array array dan integer n sebagai panjang array. Algoritma ini mulai dari elemen kedua, menganggap elemen pertama sudah terurut.
- Loop Utama (for i from 1 to n-1): Loop ini mengontrol elemen mana yang sedang diproses, mulai dari elemen kedua hingga elemen terakhir.
- Loop Pindah (while j >= 0 and array[j] > key): Elemen-elemen yang lebih besar dari key di bagian yang sudah terurut dipindahkan satu posisi ke kanan, menciptakan ruang untuk key di posisi yang tepat. j berkurang satu posisi setiap kali untuk mengecek elemen yang sebelumnya.
- Penempatan Elemen Kunci: Setelah loop pindah berakhir, key ditempatkan di posisi yang benar pada array[j + 1], menyelesaikan penempatan elemen key di dalam bagian array yang sudah terurut.
- Output Akhir: Setelah loop utama selesai, seluruh array akan terurut dalam urutan menaik

2. Rangkuman

Bubble Sort dan Insertion Sort adalah algoritma pengurutan sederhana yang menggunakan pendekatan berbeda. Bubble Sort mengurutkan data dengan menukar elemen yang bersebelahan jika diperlukan, sehingga nilai terbesar akan "menggelembung" ke posisi terakhir dalam setiap iterasi. Proses ini diulang sampai tidak ada lagi penukaran yang diperlukan. Sementara itu, Insertion Sort mengurutkan dengan menyisipkan elemen secara berurutan ke dalam posisi yang benar di bagian yang sudah terurut. Prosesnya dimulai dari elemen kedua, membandingkannya dengan elemen sebelumnya, lalu menyisipkannya di posisi yang sesuai. Algoritma ini terus diulang hingga semua elemen berada pada posisi yang benar dalam array..

3. Pustaka.

[1] Aji, Alham Fikri dan Gozali, William. Pemrograman Kompetitif Dasar. ISBN 978-602-6598-89-9.

[2] Oky Erzani, Muhammad. Algoritma Pengurutan Dalam Pemrograman.

[3] Hartanto, Willian. Implementasi Algoritma Bubble Sort dengan Bahasa Pemrograman Python.

D. Penilaian Kegiatan Belajar

1. Penilaian

1.1 Latihan

Buatlah sebuah program C++ yang mengimplementasikan algoritma Bubble Sort & Insertion Sort untuk mengurutkan array dari bilangan bulat dalam urutan menaik. Program akan

menerima input berupa jumlah elemen pada array dan elemen-elemen dari array tersebut. Setelah diurutkan, program akan mencetak array yang telah terurut.

Spesifikasi:

- Input pertama adalah integer n, yaitu jumlah elemen pada array.
- Input berikutnya adalah n bilangan bulat yang merupakan elemen-elemen dari array.
- Program harus mengurutkan array menggunakan algoritma Bubble Sort & Insertion Sort dan menampilkan array yang telah diurutkan.

Format Masukan dan Keluaran:

Input	Output
5	2 3 4 5 8
5 3 8 4 2	

1.2 Tantangan

1) Bubble Sort

<https://tlx.toki.id/courses/competitive-1/chapters/03/problems/D>

Deskripsi

Pak Dengklek memiliki setumpuk kartu yang terdiri atas N buah bilangan kartu. Kartu ke-i memiliki nomor A_i yang tercetak di atasnya.

Ia ingin mengurutkan kartu-kartu tersebut sehingga nomor-nomor pada kartu tersebut terurut tak menurun. Namun,

dalam satu langkah, Pak Dengklek hanya dapat menukar posisi dua buah kartu yang tepat bersebelahan.

Berapakah banyaknya langkah minimum yang Pak Dengklek perlukan?

Masukan

Masukan diberikan dalam format berikut:

N

A₁

A₂

:

A_N

Format Keluaran

Keluarkan sebuah baris berisi banyaknya langkah minimum yang Pak Dengklek perlukan untuk mengurutkan nomor-nomor kartu secara tak menurun.

Format masukan :

Input	Output
3	2
2	
3	
1	

Penjelasan Contoh

Pak Dengklek dapat menguratkannya dalam dua langkah:

Tukar kartu kedua dan ketiga.

Tukar kartu pertama dan kedua.

2. Umpan Balik dan Tindak Lanjut

1) Sorting

Program C++

```
#include <iostream>
using namespace std;

// Fungsi untuk mengurutkan array menggunakan
Bubble Sort

void bubbleSort(int arr[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                // Tukar jika elemen saat ini
                // lebih besar dari elemen berikutnya
                swap(arr[j], arr[j + 1]);
            }
        }
    }
}

int main() {
    int n;
```

```
// Input jumlah elemen
cout << "Masukkan jumlah elemen: ";
cin >> n;

int arr[n];

// Input elemen-elemen array
cout << "Masukkan elemen-elemen array: ";
for (int i = 0; i < n; i++) {
    cin >> arr[i];
}

// Memanggil fungsi Bubble Sort
bubbleSort(arr, n);

// Output array yang telah diurutkan
cout << "Array yang telah diurutkan: ";
for (int i = 0; i < n; i++) {
    cout << arr[i] << " ";
}
cout << endl;

return 0;
}
```

KEGIATAN BELAJAR 2

MERGE SORT & SELECTION SORT

A. Deskripsi Singkat

Materi ini dirancang untuk membantu mahasiswa memahami teknik-teknik sorting dalam algoritma, khususnya dalam mengenali algoritma merge sort dan selection sort. Pemahaman akan kedua teknik ini sangat penting, karena sorting adalah salah satu elemen inti dalam pengolahan data yang efisien dalam pemrograman. Dengan memahami kedua teknik ini, mahasiswa dapat menentukan metode sorting yang tepat sesuai kebutuhan, serta mengoptimalkan kinerja pemrograman.

Pada kegiatan belajar ini, mahasiswa akan diperkenalkan pada cara kerja merge sort dan selection sort. Dengan menguasai konsep ini, mahasiswa akan memiliki dasar logis yang kuat untuk merancang algoritma pencarian yang efisien dan sesuai konteks. Materi ini akan mencakup aturan dan langkah-langkah algoritma merge sort & selection sort dalam kedua teknik pengurutan ini, sehingga mahasiswa dapat memahami proses kerja pengurutan data secara bertahap.

B. Relevansi

Modul ini merupakan modul lanjutan dari kegiatan belajar 1 yang telah membahas mengenai konsep algoritma sorting dan beberapa algoritma sorting sebelumnya. Pada modul ini, mahasiswa akan memperluas wawasan mereka terhadap algoritma sorting dengan mempelajari dua teknik sorting lainnya, yaitu merge sort dan selection sort, dengan fokus pada cara kerja algoritma tersebut.

Sebagai modul lanjutan, materi ini dirancang untuk membantu mahasiswa tidak hanya memahami cara kerja kedua algoritma sorting ini, tetapi juga menentukan situasi yang paling tepat untuk masing-masing metode berdasarkan kebutuhan dan jenis data yang diolah.

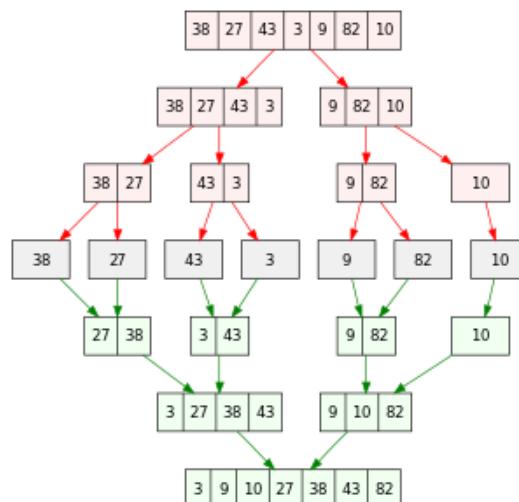
Dalam rangkaian modul ini, pemahaman tentang algoritma sorting dengan menggunakan kedua teknik merge sort dan selection sort sangat relevan terhadap pencapaian Sub-Capaian Pembelajaran Mata Kuliah (Sub-CPMK), karena memberikan dasar analitis bagi mahasiswa untuk dapat merancang algoritma secara logis dan efisien. Dengan pemahaman ini, akan membantu mahasiswa mengembangkan keterampilan dalam merancang algoritma yang akan diimplementasikan dalam berbagai bahasa pemrograman pada modul-modul selanjutnya.

C. Pembelajaran

Capaian pembelajaran yang diharapkan pada kegiatan belajar yang pertama ini adalah peserta kuliah akan mampu mengetahui dasar algoritma. Untuk mencapai kompetensi lulusan ini, peserta kuliah diharapkan membaca dan menyelesaikan Kegiatan Belajar 1 ini. Untuk mencapai capaian pembelajaran pada Kegiatan Belajar 1, peserta kuliah harus mengikuti urutan kegiatan pembelajaran yang telah disiapkan dalam modul ini.

1. Uraian

1.1 Merge Sort



Merge sort adalah algoritma pengurutan yang memiliki kompleksitas $O(N \log N)$. Algoritma merge sort terus membagi array yang ingin diurutkan menjadi dua sampai tersisa satu elemen, kemudian menggabungkannya kembali sambil mengurutkannya. Inti pengurutan merge sort ada pada tahap combine. Cara kerja algoritma ini adalah:

1. Divide : jika array yang akan diurutkan lebih dari 1 elemen, bagi array tersebut menjadi dua array sama besar (atau mendekati sama besar jika panjang array tersebut ganjil). Kemudian lakukan merge sort pada masing-masing subarray tersebut.
2. Conquer: ketika array berisi hanya 1 elemen, tidak perlu lakukan apapun. Array yang berisi 1 elemen sudah pasti terurut.
3. Combine : saat kita memiliki dua array yang telah terurut, kita bisa menggabungkan (merge) keduanya menjadi sebuah array yang terurut..

Contoh Kode Algoritma Merge Sort

```
#include <iostream>

using namespace std;

void merge(int arr[], int left, int mid, int right) {

    int n1 = mid - left + 1;
    int n2 = right - mid;
```

```
// Buat array sementara  
  
int L[n1], R[n2];  
  
// Copy data ke array sementara L dan R  
  
for (int i = 0; i < n1; i++)  
    L[i] = arr[left + i];  
  
for (int j = 0; j < n2; j++)  
    R[j] = arr[mid + 1 + j];  
  
// Gabungkan kedua array kembali ke arr  
  
int i = 0, j = 0, k = left;  
  
while (i < n1 && j < n2) {  
    if (L[i] <= R[j]) {  
        arr[k] = L[i];  
        i++;  
    } else {  
        arr[k] = R[j];  
        j++;  
    }  
    k++;  
}  
  
// Copy elemen sisanya (jika ada)  
  
while (i < n1) {  
    arr[k] = L[i];
```

```
i++;
k++;
}

while (j < n2) {
    arr[k] = R[j];
    j++;
    k++;
}

}

void mergeSort(int arr[], int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;

        // Panggil mergeSort untuk setengah
        bagian kiri
        mergeSort(arr, left, mid);

        // Panggil mergeSort untuk setengah
        bagian kanan
        mergeSort(arr, mid + 1, right);

        // Gabungkan kedua bagian yang telah
        terurut
        merge(arr, left, mid, right);
    }
}
```

```
}
```

```
void printArray(int A[], int size) {
    for (int i = 0; i < size; i++)
        cout << A[i] << " ";
}

int main() {
    int arr[] = {38, 27, 43, 3, 9, 82, 10};
    int arr_size = sizeof(arr) / sizeof(arr[0]);

    mergeSort(arr, 0, arr_size - 1);
    printArray(arr, arr_size);
    return 0;
}
```

Output : 3 9 10 27 38 43 82

Pseudocode Algoritma Merge Sort

```
1. FUNCTION mergeSort(arr, left, right)
    1.1 IF left < right
        1.2 mid = left + (right - left) / 2
        1.3 mergeSort(arr, left, mid)
        // Rekursif untuk bagian kiri
        1.4 mergeSort(arr, mid + 1, right)
        // Rekursif untuk bagian kanan
        1.5 merge(arr, left, mid, right)
        // Gabungkan dua bagian yang sudah terurut
```

```
2. FUNCTION merge(arr, left, mid, right)
    2.1 n1 = mid - left + 1
    // Jumlah elemen di bagian kiri
    2.2 n2 = right - mid
    // Jumlah elemen di bagian kanan
    2.3 Buat array sementara L dan R dengan
        ukuran n1 dan n2

    2.4 Copy elemen dari arr[left...mid] ke L
    2.5 Copy elemen dari arr[mid+1...right] ke R

    2.6 Inisialisasi i = 0, j = 0, k = left
    2.7 WHILE i < n1 AND j < n2
        2.7.1 IF L[i] <= R[j]
            arr[k] = L[i]
            i++
        ELSE
            arr[k] = R[j]
            j++
        k++

    2.8 Copy sisa elemen di L (jika ada) ke arr
    2.9 Copy sisa elemen di R (jika ada) ke arr

3. FUNCTION printArray(arr, size)
    3.1 Print elemen array arr dari indeks 0
        hingga size-1
```

4. START

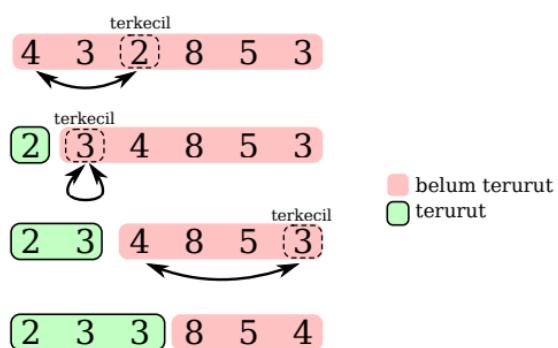
4.1 Buat array arr

4.2 Panggil mergeSort(arr, 0, panjang(arr) - 1)

4.3 Print array yang sudah diurutkan menggunakan printArray

1. Array diberikan ke fungsi mergeSort.
2. Array akan dipecah menjadi dua bagian secara rekursif.
3. Setelah array terpecah sampai elemen individu, fungsi merge akan menggabungkannya satu per satu hingga array kembali menjadi satu kesatuan yang sudah terurut.

1.2 Selection Sort



Selection Sort adalah algoritma pengurutan berbasis perbandingan. Algoritma ini mengurutkan array dengan cara berulang kali memilih elemen terkecil (atau terbesar) dari bagian yang tidak diurutkan dan menukarnya dengan elemen pertama yang tidak diurutkan. Proses ini berlanjut hingga seluruh array diurutkan.

Berikut adalah langkah-langkah untuk melakukan selection sort:

1. Pilih elemen terkecil dari data, lalu pindahkan ke elemen pertama.
2. Pilih elemen terkecil dari data yang tersisa, lalu pindahkan ke elemen kedua.
3. Pilih elemen terkecil dari data yang tersisa, lalu pindahkan ke elemen ketiga.
4. dan seterusnya sampai seluruh elemen terurut.

Contoh Kode Algoritma Selection Sort

```
#include <iostream>

using namespace std;

// Fungsi untuk melakukan selection sort
void selectionSort(int arr[], int n) {
    for (int i = 0; i < n - 1; i++) {
        // Anggap elemen pertama adalah yang
        terkecil
        int minIndex = i;

        // Temukan indeks elemen terkecil dalam
        sisa array
        for (int j = i + 1; j < n; j++) {
            if (arr[j] < arr[minIndex]) {
                minIndex = j; // Update indeks
                terkecil
            }
        }
    }
}
```

```
// Tukar elemen terkecil dengan elemen
pertama

swap(arr[i], arr[minIndex]);

}

}

int main() {
    int arr[] = {4, 3, 2, 8, 5, 3}; // Nilai
array yang diperbarui

    int n = sizeof(arr) / sizeof(arr[0]);

        selectionSort(arr, n); // Panggil
selectionSort untuk mengurutkan array

        // Cetak array yang telah diurutkan
        for (int i : arr) {
            cout << i << " ";
        }
        cout << endl;

    return 0;
}}
```

Output : 2 3 3 4 5 8

Pseudocode Algoritma Insertion Sort

1. FUNCTION selectionSort(arr, n)

 1.1 FOR i = 0 to n - 2

```
1.2 minIndex = i // Anggap elemen pertama sebagai yang terkecil
```

```
1.3 FOR j = i + 1 to n - 1
```

```
    1.3.1 IF arr[j] < arr[minIndex]
```

```
        minIndex = j // Update indeks elemen terkecil
```

```
1.4 Tukar elemen arr[i] dengan arr[minIndex] // Tukar elemen pertama dengan elemen terkecil
```

```
2. FUNCTION printArray(arr, n)
```

```
    2.1 FOR setiap elemen i di arr
```

```
        Print elemen i
```

```
3. START
```

```
    3.1 Buat array arr dan tentukan ukurannya n
```

```
    3.2 Panggil selectionSort(arr, n) untuk mengurutkan array
```

```
    3.3 Panggil printArray(arr, n) untuk mencetak array yang telah diurutkan
```

1. Inisialisasi dan Looping:

Algoritma Selection Sort melakukan iterasi dari indeks pertama hingga satu elemen sebelum terakhir. Dalam setiap iterasi, algoritma memilih elemen terkecil dari sisa elemen array dan menukarnya dengan elemen di posisi awal iterasi.

2. Menemukan Elemen Terkecil:

Pada setiap iterasi i , algoritma mengasumsikan bahwa elemen di posisi i adalah yang terkecil di sisa array. Kemudian, ia membandingkan elemen ini dengan elemen-elemen berikutnya (j dari $i + 1$ hingga $n - 1$). Jika ditemukan elemen yang lebih kecil, minIndex diperbarui ke posisi elemen yang lebih kecil tersebut.

3. Penukaran:

Setelah menemukan elemen terkecil, algoritma akan menukarinya dengan elemen di posisi i (jika elemen terkecil tidak berada di posisi i). Dengan demikian, elemen di posisi i sudah berada di posisi yang benar dalam urutan.

4. Mengulangi Proses:

Proses di atas diulangi untuk setiap elemen, hingga array menjadi terurut.

5. Output:

Setelah semua iterasi selesai, array menjadi terurut dalam urutan menaik.

Contoh Eksekusi dengan Array {4, 3, 2, 8, 5, 3}

Iterasi 1 (i=0): Temukan elemen terkecil di {4, 3, 2, 8, 5, 3}, yaitu 2. Tukar 2 dengan 4.

Array: {2, 3, 4, 8, 5, 3}

Iterasi 2 (i=1): Temukan elemen terkecil di {3, 4, 8, 5, 3}, yaitu 3. Tidak perlu menukar.

Array: {2, 3, 4, 8, 5, 3}

Iterasi 3 (i=2): Temukan elemen terkecil di {4, 8, 5, 3}, yaitu 3.
Tukar 3 dengan 4.

Array: {2, 3, 3, 8, 5, 4}

Iterasi 4 (i=3): Temukan elemen terkecil di {8, 5, 4}, yaitu 4.
Tukar 4 dengan 8.

Array: {2, 3, 3, 4, 5, 8}

2. Rangkuman

Merge Sort adalah algoritma pengurutan dengan kompleksitas $O(N \log N)$ yang menggunakan pendekatan divide and conquer. Algoritma ini membagi array menjadi dua bagian hingga setiap bagian hanya memiliki satu elemen, kemudian menggabungkannya kembali dengan cara mengurutkan elemen-elemen tersebut. Merge Sort efisien untuk data besar, namun memerlukan ruang tambahan karena penggabungan.

Selection Sort adalah algoritma pengurutan berbasis perbandingan dengan kompleksitas $O(N^2)$. Algoritma ini mengurutkan array dengan memilih elemen terkecil dari bagian yang tidak terurut dan menukarinya dengan elemen pertama yang tidak terurut. Meskipun sederhana, Selection Sort kurang efisien untuk data besar dan tidak stabil.

3. Pustaka.

[1] Aji, Alham Fikri dan Gozali, William. Pemrograman Kompetitif Dasar. ISBN 978-602-6598-89-9.

[2] Oky Erzani, Muhammad. Algoritma Pengurutan Dalam Pemrograman.

D. Penilaian Kegiatan Belajar

1. Penilaian

1.1 Latihan

Diberikan sebuah array yang berisi sejumlah angka yang belum terurut. Tugas Anda adalah untuk mengurutkan array tersebut menggunakan algoritma Merge Sort atau Selection Sort dan mencetak hasilnya.

Deskripsi Input:

Sebuah array yang berisi angka bulat dengan panjang n ($1 \leq n \leq 1000$) yang belum terurut.

Deskripsi Output:

Cetak array setelah diurutkan dalam urutan menaik.

Format Masukan dan Keluaran:

Input	Output
6 4 3 2 8 5 3	2 3 3 4 5 8
5 12 9 5 15 1	1 5 9 12 15

1.2 Tantangan

- 1) Bubble Sort

<https://tlx.toki.id/courses/competitive-1/chapters/03/problems/C>

Deskripsi

Pak Dengklek memiliki data yang terdiri atas N buah kata. Ia ingin mengurutkan data tersebut. Bantulah ia!

Dalam pengurutannya, kata-kata diurutkan berdasarkan banyaknya hurufnya, dari yang paling sedikit ke yang paling banyak. Jika banyak hurufnya sama, maka kemudian diurutkan secara leksikografis (urutan kamus).

Masukan

Baris pertama berisi sebuah bilangan bulat N. N baris berikutnya masing-masing berisi sebuah kata.

Format Keluaran

N buah baris berisi kata-kata yang sudah diurutkan berdasarkan aturan pada deskripsi.

Format masukan :

Input	Output
6	man
angela	budi
budi	didi
didi	sani
man	andri
andri	angela
sani	

2. Umpan Balik dan Tindak Lanjut

1) Merge & Selection Sort

Program C++ Merge Sort

```
#include <iostream>

using namespace std;

// Fungsi untuk menggabungkan dua bagian array
void merge(int arr[], int left, int mid, int right) {

    int n1 = mid - left + 1;
    int n2 = right - mid;

    // Buat array sementara
    int L[n1], R[n2];

    // Copy data ke array sementara L dan R
    for (int i = 0; i < n1; i++)
        L[i] = arr[left + i];
    for (int j = 0; j < n2; j++)
        R[j] = arr[mid + 1 + j];

    // Gabungkan kedua array kembali ke arr
    int i = 0, j = 0, k = left;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        }
        else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    // Jika ada angka yang belum ditulis
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}
```

```
        } else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    // Copy elemen sisanya (jika ada)
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}

// Fungsi mergeSort untuk mengurutkan array
void mergeSort(int arr[], int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;
```

```
// Panggil mergeSort untuk setengah  
bagian kiri  
  
mergeSort(arr, left, mid);  
  
// Panggil mergeSort untuk setengah  
bagian kanan  
  
mergeSort(arr, mid + 1, right);  
  
// Gabungkan kedua bagian yang telah  
terurut  
  
merge(arr, left, mid, right);  
}  
}  
  
int main() {  
    int n;  
  
    cout << "Masukkan jumlah elemen array: ";  
    cin >> n;  
  
    int arr[n];  
  
    cout << "Masukkan elemen array: ";  
    for (int i = 0; i < n; i++) {  
        cin >> arr[i];  
    }  
  
    mergeSort(arr, 0, n - 1); // Panggil  
    mergeSort untuk mengurutkan array
```

```
// Cetak array yang telah diurutkan
cout << "Array yang terurut: ";
for (int i = 0; i < n; i++) {
    cout << arr[i] << " ";
}
cout << endl;

return 0;
}
```

Program C++ Selection Sort

```
#include <iostream>
using namespace std;

// Fungsi untuk melakukan selection sort
void selectionSort(int arr[], int n) {
    for (int i = 0; i < n - 1; i++) {
        // Anggap elemen pertama adalah yang
        terkecil
        int minIndex = i;

        // Temukan indeks elemen terkecil dalam
        sisa array
    }
}
```

```
        for (int j = i + 1; j < n; j++) {  
            if (arr[j] < arr[minIndex]) {  
                minIndex = j; // Update indeks  
                terkecil  
            }  
        }  
  
        // Tukar elemen terkecil dengan elemen  
        pertama  
        swap(arr[i], arr[minIndex]);  
    }  
}  
  
int main() {  
    int n;  
    cout << "Masukkan jumlah elemen array: ";  
    cin >> n;  
  
    int arr[n];  
    cout << "Masukkan elemen array: ";  
    for (int i = 0; i < n; i++) {  
        cin >> arr[i];  
    }  
  
    selectionSort(arr, n); // Panggil  
    selectionSort untuk mengurutkan array
```

```
// Cetak array yang telah diurutkan
cout << "Array yang terurut: ";
for (int i = 0; i < n; i++) {
    cout << arr[i] << " ";
}
cout << endl;

return 0;
}
```

KEGIATAN BELAJAR 3

QUICK SORT & KOMPLEKSITAS ALGORITMA

A. Deskripsi Singkat

Materi ini dirancang untuk membantu mahasiswa memahami teknik sorting dalam algoritma, khususnya dalam mengenali algoritma quick sort dan menganalisa kompleksitasnya. Pemahaman akan algoritma ini sangat penting, karena sorting adalah salah satu elemen inti dalam pengolahan data yang efisien dalam pemrograman. Dengan memahami kedua teknik ini, mahasiswa dapat menentukan metode sorting yang tepat sesuai kebutuhan, serta mengoptimalkan kinerja pemrograman.

Pada kegiatan belajar ini, mahasiswa akan diperkenalkan pada cara kerja quick sort. Dengan menguasai konsep ini, mahasiswa akan memiliki dasar logis yang kuat untuk merancang algoritma sorting yang efisien dan sesuai konteks. Materi ini akan mencakup aturan dan langkah-langkah algoritma quick sort, sehingga mahasiswa dapat memahami proses kerja pengurutan data dan kompleksitasnya.

B. Relevansi

Modul ini merupakan modul lanjutan dari kegiatan belajar 2 yang telah membahas mengenai konsep algoritma sorting dan beberapa algoritma sorting sebelumnya. Pada modul ini, mahasiswa akan memperluas wawasan mereka terhadap algoritma sorting dengan mempelajari teknik sorting lainnya, yaitu quick sort, dengan fokus pada cara kerja algoritma tersebut dan kompleksitasnya.

Sebagai modul lanjutan, materi ini dirancang untuk membantu mahasiswa tidak hanya memahami cara kerja algoritma quick sort ini, tetapi juga menentukan situasi yang paling tepat untuk masing-masing metode berdasarkan kebutuhan dan jenis data yang diolah.

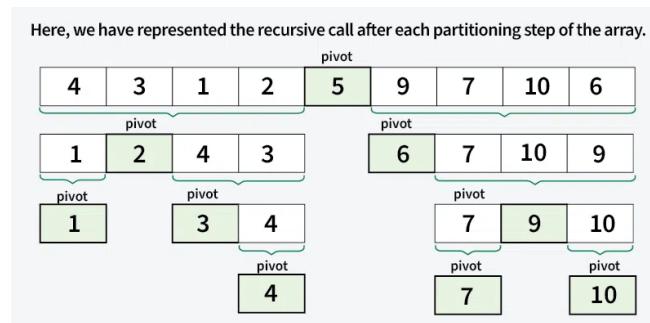
Dalam rangkaian modul ini, pemahaman tentang algoritma sorting dengan menggunakan teknik quick sort sangat relevan terhadap pencapaian Sub-Capaian Pembelajaran Mata Kuliah (Sub-CPMK), karena memberikan dasar analitis bagi mahasiswa untuk dapat merancang algoritma secara logis dan efisien. Dengan pemahaman ini, akan membantu mahasiswa mengembangkan keterampilan dalam merancang algoritma yang akan diimplementasikan dalam berbagai bahasa pemrograman pada modul-modul selanjutnya.

C. Pembelajaran

Capaian pembelajaran yang diharapkan pada kegiatan belajar yang pertama ini adalah peserta kuliah akan mampu mengetahui dasar algoritma. Untuk mencapai kompetensi lulusan ini, peserta kuliah diharapkan membaca dan menyelesaikan Kegiatan Belajar 1 ini. Untuk mencapai capaian pembelajaran pada Kegiatan Belajar 1, peserta kuliah harus mengikuti urutan kegiatan pembelajaran yang telah disiapkan dalam modul ini.

1. Uraian

1.1 Quick Sort



Terdapat algoritma pengurutan selain beberapa algoritma sorting yang telah dijelaskan pada kegiatan sebelumnya, salah satunya adalah quicksort . Quicksort menggunakan prinsip divide and

conquer dalam pengurutannya. Tahapan dari quicksort adalah sebagai berikut:

1. Divide : pilih suatu elemen yang kita sebut pivot, kemudian kita bagi array menjadi dua sehingga salah satunya selalu \leq pivot dan yang sisanya selalu $>$ pivot.

Kemudian, lakukan quicksort pada masing-masing subarray tersebut.

2. Conquer: ketika array hanya memiliki satu elemen, array tersebut sudah terurut.
3. Combine : gabungkan subarray dengan menempelkan hasil quicksort bagian kiri dan kanan.

Bagian utama dari quicksort adalah proses partisi (bagian divide). Sebelum melakukan partisi, pilih satu elemen yang akan dijadikan pivot. Kemudian, lakukan partisi supaya seluruh elemen yang \leq pivot berada di sebelah kiri dari pivot, dan yang $>$ pivot di sebelah kanannya. Pemilihan elemen pivot dapat dilakukan secara bebas. Bahkan pemilihan pivot dengan cara tertentu dapat mempengaruhi performa algoritma quicksort. Pilihan yang cukup sering dipakai adalah menggunakan elemen di tengah array sebagai pivot.

Contoh Kode Algoritma Merge Sort

```
#include <bits/stdc++.h>
using namespace std;

int partition(vector<int>& arr, int low, int high) {
```

```
// Choose the pivot  
  
int pivot = arr[high];  
  
// Index of smaller element and indicates  
// the right position of pivot found so far  
int i = low - 1;  
  
// Traverse arr[low..high] and move all  
smaller  
// elements on left side. Elements from low  
to  
// i are smaller after every iteration  
for (int j = low; j <= high - 1; j++) {  
    if (arr[j] < pivot) {  
        i++;  
        swap(arr[i], arr[j]);  
    }  
}  
  
// Move pivot after smaller elements and  
// return its position  
swap(arr[i + 1], arr[high]);  
return i + 1;  
}  
  
// The QuickSort function implementation
```

```
void quickSort(vector<int>& arr, int low, int
high) {

    if (low < high) {

        // pi is the partition return index of
        pivot
        int pi = partition(arr, low, high);

        // Recursion calls for smaller elements
        // and greater or equals elements
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

int main() {
    vector<int> arr = {10, 7, 8, 9, 1, 5};
    int n = arr.size();
    quickSort(arr, 0, n - 1);

    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    return 0;
}
```

Output : 1 5 7 8 9 10

Pseudocode Algoritma Merge Sort

```
QuickSort(arr, low, high)
    jika low < high maka
        pi = Partition(arr, low, high) // Temukan indeks pivot
        QuickSort(arr, low, pi - 1) // Sortir bagian kiri secara rekursif
        QuickSort(arr, pi + 1, high) // Sortir bagian kanan secara rekursif

Partition(arr, low, high)
    pivot = arr[high] // Pilih elemen terakhir sebagai pivot
    i = low - 1 // Inisialisasi indeks elemen yang lebih kecil
    untuk j = low sampai high - 1 lakukan
        jika arr[j] < pivot maka // Jika elemen saat ini lebih kecil dari pivot
            i = i + 1
            tukar arr[i] dengan arr[j] // Tukar elemen untuk menempatkan elemen lebih kecil di kiri
            tukar arr[i + 1] dengan arr[high] // Pindahkan pivot ke posisi yang benar
    kembalikan i + 1 // Kembalikan indeks pivot
```

1. Fungsi QuickSort adalah algoritma pengurutan rekursif yang bekerja dengan membagi array menjadi bagian-bagian yang lebih kecil.
 - Fungsi ini menerima sebuah array (arr), serta dua indeks, low dan high, yang mendefinisikan bagian array yang akan diurutkan.
 - Jika low lebih kecil dari high, fungsi akan memanggil fungsi Partition untuk menemukan posisi yang benar dari pivot.
 - Setelah partitioning, array dibagi menjadi dua subarray, dan masing-masing disortir secara rekursif.
2. Fungsi Partition mengatur elemen-elemen di dalam array sedemikian rupa sehingga elemen yang lebih kecil dari pivot berada di sisi kiri, dan elemen yang lebih besar berada di sisi kanan.
 - Elemen terakhir ($arr[high]$) dipilih sebagai pivot.
 - Variabel i adalah indeks elemen yang lebih kecil, diinisialisasi dengan $low - 1$. Variabel j digunakan untuk melakukan iterasi pada array.
 - Jika elemen $arr[j]$ lebih kecil dari pivot, i akan meningkat dan elemen $arr[i]$ ditukar dengan elemen $arr[j]$.
 - Setelah selesai melakukan iterasi, pivot akan dipindahkan ke posisi yang benar dengan menukar elemen $arr[i + 1]$ dan $arr[high]$.
 - Fungsi Partition kemudian mengembalikan indeks pivot yang digunakan oleh fungsi QuickSort untuk menyortir subarray sebelah kiri dan kanan pivot.

Contoh Proses QuickSort: Untuk array {10, 7, 8, 9, 1, 5}:

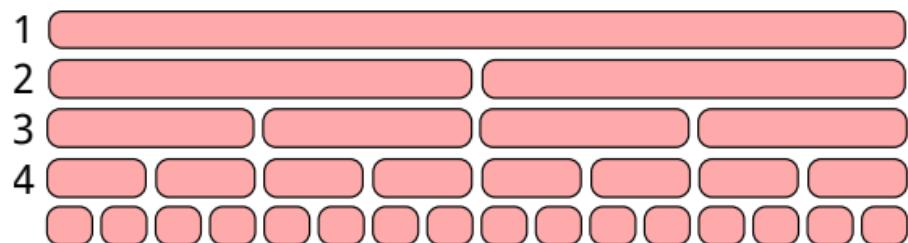
1. Pivot yang dipilih adalah 5 (elemen terakhir).

2. Fungsi Partition menempatkan 5 pada posisi yang benar di dalam array yang sudah terurut: {1, 5, 8, 9, 7, 10}.
3. Sekarang, array terbagi menjadi dua subarray: {1} dan {8, 9, 7, 10}.
4. Proses ini kemudian diulang untuk kedua subarray tersebut:
5. Untuk {8, 9, 7, 10}, pivot 10 dipindahkan ke posisi yang benar, dan subarray {8, 9, 7} disortir lebih lanjut.
6. Begitu juga, {8, 9, 7} disortir dengan memindahkan 7 ke posisi yang benar.
7. Hasil akhirnya adalah array yang terurut: {1, 5, 7, 8, 9, 10}.

1.2 Kompleksitas Algoritma

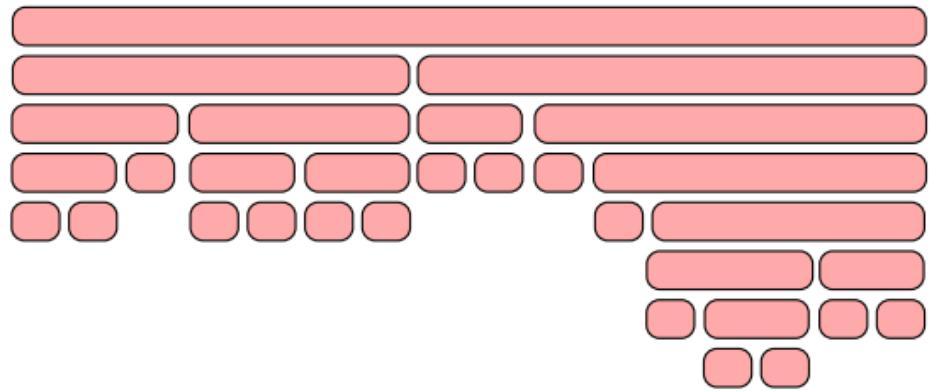
Pada setiap kedalaman rekursi, array hasil partisi belum tentu memiliki ukuran yang sama. Hasil partisi bergantung pada nilai pivot yang kita pilih. Karena hal ini, kita dapat menganalisis kompleksitas quicksort dalam 3 kasus:

1. Best Case



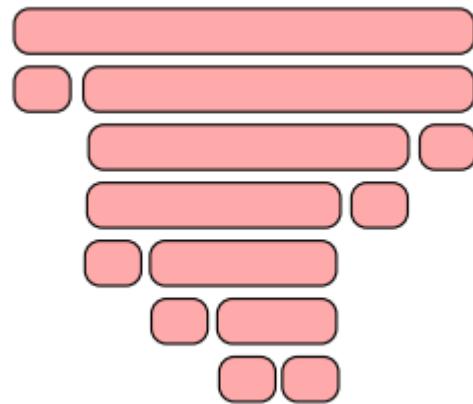
Pembelahan menjadi dua subarray sama besar menjamin kedalaman rekursif sedangkan mungkin. Sehingga untuk kasus terbaik, jalannya algoritma menjadi seperti merge sort dan bekerja dalam $O(N \log N)$.

2. Average Case



Pada kebanyakan kasus, ukuran hasil partisi berbeda-beda. Secara rata-rata kompleksitasnya masih dapat dianggap $O(N \log N)$.

3. Worst Case



Kasus paling buruk: ukuran hasil partisi sangat timpang. Hal ini terjadi ketika partisi membagi array berukuran N menjadi dua subarray yang masing-masing berukuran $N - 1$ dan 1. Dalam kasus ini, kedalaman rekursif menjadi mendekati N yang mengakibatkan kompleksitas total quicksort menjadi $O(N^2)$.

2. Rangkuman

Merge Sort adalah algoritma pengurutan dengan kompleksitas $O(N \log N)$ yang menggunakan pendekatan divide and conquer.



Algoritma ini membagi array menjadi dua bagian hingga setiap bagian hanya memiliki satu elemen, kemudian menggabungkannya kembali dengan cara mengurutkan elemen-elemen tersebut. Merge Sort efisien untuk data besar, namun memerlukan ruang tambahan karena penggabungan.

Selection Sort adalah algoritma pengurutan berbasis perbandingan dengan kompleksitas $O(N^2)$. Algoritma ini mengurutkan array dengan memilih elemen terkecil dari bagian yang tidak terurut dan menukarinya dengan elemen pertama yang tidak terurut. Meskipun sederhana, Selection Sort kurang efisien untuk data besar dan tidak stabil.

3. Pustaka.

- [1] Aji, Alham Fikri dan Gozali, William. Pemrograman Kompetitif Dasar. ISBN 978-602-6598-89-9.
- [2] Oky Erzani, Muhammad. Algoritma Pengurutan Dalam Pemrograman.

D. Penilaian Kegiatan Belajar

1. Penilaian

1.1 Latihan

Diberikan sebuah array yang berisi angka-angka acak, tugas Anda adalah mengurutkan array tersebut menggunakan algoritma QuickSort dan menampilkan hasil array yang sudah terurut.

Deskripsi Input:

Sebuah array dengan elemen-elemen angka integer.

Deskripsi Output:

Array yang telah diurutkan dari yang terkecil hingga yang terbesar.

Format Masukan dan Keluaran:

Input	Output
10, 7, 8, 9, 1, 5	1 5 7 8 9 10

1.2 Tantangan

- 1) <https://tlx.toki.id/courses/competitive-1/chapters/05/problems/A>

Deskripsi

Anda diberikan dua buah bilangan bulat A dan B, hitunglah nilai dari A^B . Agar lebih menantang, perhatikan juga format keluaran di bawah ini!.

Batasan

$1 \leq A, B \leq 10000000000$

Masukan

Sebuah baris berisi dua buah bilangan bulat A dan B.

Format Keluaran

Sebuah baris berisi nilai dari A^B . Jika bilangan tersebut lebih besar dari 999999, cukup cetak 6 digit terakhir dari bilangan tersebut..

Format masukan :

Input	Output
3 8	6561

9 9	420489
10 10	000000

2. Umpan Balik dan Tindak Lanjut

1) Quick Sort

Program C++ Quick Sort

```
#include <bits/stdc++.h>

using namespace std;

// Fungsi Partition untuk mengatur posisi pivot
int partition(vector<int>& arr, int low, int
high) {
    // Memilih pivot, yang di sini adalah elemen
    // terakhir
    int pivot = arr[high];

    int i = low - 1; // Indeks elemen yang
    // lebih kecil

    // Traverse array dan tempatkan elemen yang
    // lebih kecil dari pivot di sebelah kiri
    for (int j = low; j <= high - 1; j++) {
        if (arr[j] < pivot) {
            i++;
            swap(arr[i], arr[j]);
        }
    }
}
```

```
// Pindahkan pivot ke posisi yang benar
swap(arr[i + 1], arr[high]);
return i + 1;
}

// Fungsi QuickSort untuk menyortir array secara
rekursif

void quickSort(vector<int>& arr, int low, int
high) {
    if (low < high) {
        // pi adalah indeks pivot yang sudah
        terurut
        int pi = partition(arr, low, high);

        // Rekursi pada bagian kiri dan kanan
        pivot
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

int main() {
    vector<int> arr = {10, 7, 8, 9, 1, 5}; // 
    Array yang diberikan
    int n = arr.size();

    // Menampilkan array sebelum disortir
    cout << "Array sebelum disortir: ";
```

```
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;

    // Menyortir array menggunakan QuickSort
    quickSort(arr, 0, n - 1);

    // Menampilkan array setelah disortir
    cout << "Array setelah disortir: ";
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
    return 0;
}
```



ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA

MODUL IX

STRUCT DAN LINKED LIST



*Elly Warni
Ingrid Nurtanio
Amil Ahmad Ilham
Adnan
Anugrayani Bustamin*

PRAKATA

Alhamdulillah, puji dan syukur kami panjatkan ke hadirat Allah SWT yang telah melimpahkan rahmat dan hidayah-Nya, sehingga modul ini dapat terselesaikan dengan baik. Shalawat serta salam senantiasa tercurah kepada junjungan kita, Nabi Muhammad SAW, beserta keluarga dan sahabat beliau yang telah membawa kita dari zaman kegelapan menuju terang benderang ilmu pengetahuan.

Modul ini disusun sebagai bagian dari rangkaian pembelajaran yang bertujuan untuk membantu mahasiswa memahami konsep dasar struktur data struct dan linked list, mengimplementasikannya dalam kode C++.

Tidak dapat dipungkiri bahwa proses penyusunan modul ini tidak lepas dari dukungan berbagai pihak. Oleh karena itu, kami mengucapkan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan masukan dan bimbingan dalam proses penyusunan modul ini. Terima kasih kami sampaikan kepada rekan-rekan dosen dan staf yang telah memberikan kritik dan saran yang membangun, sehingga modul ini dapat terselesaikan dengan baik.

Kami menyadari bahwa modul ini masih memiliki keterbatasan dan kekurangan. Oleh karena itu, kami sangat mengharapkan masukan, kritik, dan saran dari para pembaca untuk penyempurnaan di masa yang akan datang. Semoga modul ini dapat memberikan manfaat dan menjadi bahan referensi yang berharga bagi para mahasiswa dalam memahami konsep algoritma dan pemrograman.

Akhir kata, semoga upaya ini mendapat ridha Allah SWT dan memberikan kontribusi positif bagi kemajuan ilmu pengetahuan. Amin.



Makassar, November 2024

Penulis

DAFTAR ISI

PRAKATA	i
DAFTAR ISI	iii
RPS MATA KULIAH ALGORITMA DAN STRUKTUR DATA	iv
MODUL ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA	10
Pengantar	10
KEGIATAN BELAJAR 1	12
A. Deskripsi Singkat	12
B. Relevansi	13
C. Pembelajaran	14
D. Penilaian Kegiatan Belajar	20

RPS MATA KULIAH ALGORITMA DAN STRUKTUR DATA



**UNIVERSITAS HASANUDDIN
FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK INFORMATIKA**

Kode Dokumen

RENCANA PEMBELAJARAN SEMESTER

MATA KULIAH (MK)	KODE	Rumpun MK	BOBOT (sks)		SEMESTER	Tgl Penyusunan				
Algoritma Pemrograman dan Struktur Data	23D12110604	Teknik Informatika	T=2	P=2	2	19-02-2023				
OTORISASI		Pengembang RPS Dr. Ir. Ingrid Nurtanio., MT Dr. Amil Ahmad Ilham., ST., M.IT Adnan., ST., MT., PhD Anugrayani Bustamin., ST., MT Elly Warni, S.T., M.T	Koordinator RMK Dr. Ir. Ingrid Nurtanio., MT		Ketua PRODI Prof. Dr. Indrabayu., ST., M.T., M.Bus.Sys					
Capaian Pembelajaran (CP)	CPL-PRODI yang dibebankan pada MK <i>Intended Learning Outcomes</i>									
	CPL 1	Memiliki dasar pengetahuan Teknik Informatika yang meliputi teori dan konsep dasar dari Ilmu Komputer, Matematika dan Statistika, Algoritma dan Pemrograman, Rekayasa Perangkat Lunak, Manajemen Informasi dan Ketahanan Digital, serta pengetahuan tingkat lanjut pada bidang-bidang khusus Teknik Informatika, seperti Kecerdasan Buatan, Data Science, Jaringan Komputer, Komputasi Awan dan Internet of Things.								
	CPL 3	Mampu mengaplikasikan pengetahuan bidang Teknik Informatika yang dipadankan dengan bidang ilmu lainnya untuk menganalisa dan mencari solusi dari berbagai masalah berbasis komputasi.								
	CPL 4	Mampu mendesain, mengimplementasikan dan mengevaluasi solusi berbasis komputasi dengan mengaplikasikan ilmu Teknik Informatika dan dasar-dasar pembangunan perangkat lunak.								
	Capaian Pembelajaran Mata Kuliah (CPMK)									
	CPMK	Setelah mengikuti Mata Kuliah Algoritma Pemrograman dan Struktur Data selama 1 (satu) semester, mahasiswa mampu menguasai dan menerapkan konsep teoritis bidang Informatika dan ilmu komputer secara dan mendalam serta memformulasikan penyelesaian masalah procedural dengan pemikiran logis dan sistematis secara mandiri dan terukur sebagai dasar pengembangan perangkat lunak.								
	CPL ⇒ Sub-CPMK									
	CPL 1	Sub-CPMK 1 : Mahasiswa mampu mendefinisikan konsep dan pengertian Algoritma								

	CPL 1	Sub-CPMK 2 : Mahasiswa mampu mendeksripsikan Dasar Algoritma
	CPL 3	Sub-CPMK 3 : Mahasiswa mampu mengimplementasikan Flowchart
	CPL 1	Sub-CPMK 4 : Mahasiswa mampu memahami bentuk pemilihan dan perulangan
	CPL 4	Sub-CPMK 5 : Mahasiswa mampu mengimplementasikan prosedur, fungsi dan pemrosesan teks
	CPL 1	Sub-CPMK 6 : Mahasiswa mampu mempraktekkan konsep array
	CPL 4	Sub-CPMK 7 : Mahasiswa mampu menetapkan ciri dari teknik pencarian
	CPL 4	Sub-CPMK 8 : Mahasiswa mampu menetapkan ciri dari teknik sorting dan menganalisis complexitas algoritma
	CPL 1	Sub-CPMK 9 : Mahasiswa mampu memahami struct dan konsep linked list
	CPL 1	Sub-CPMK 10 : Mahasiswa mampu memahami Antrian dan Stack
	CPL 1	Sub-CPMK 11 : Mahasiswa mampu memahami teknik graph
	CPL 3	Sub-CPMK 12 : Mahasiswa mampu mengimplementasi konsep Tree
Deskripsi Singkat MK	<p>Mata kuliah Algoritma Pemrograman dan Struktur Data merupakan mata kuliah yang menyajikan konsep pemrograman secara mendalam terkait algoritma dan struktur data. Muatan mata kuliah merupakan dasar dari ilmu matematika, algoritma dan logika yang menjadi syarat utama pada mata kuliah Pemrograman lanjut . Mata kuliah ini disajikan untuk mahasiswa semester dua di Program Studi Teknik Informatika, Penanggungjawab mata kuliah adalah Laboratorium Komputer . Metode Pembelajaran bauran, teori dan praktik</p>	
Bahan Kajian / Materi Pembelajaran	<ol style="list-style-type: none"> 1. Konsep Algoritma 2. Dasar Algoritma (Aturan Penulisan Header, deklarasi dan deskripsi) 3. Flowchart (Simbol, variabel, algoritma cabang, runtunan) 4. Pemilihan (Alir Kontrol dan Pengulangan) 5. Prosedur, Fungsi dan Pemrosesan Teks 6. Array – Larik 7. Pencarian (Binary Search) 8. Pencarian (Sequential Search) 9. Bubble dan Insertion Sort 10. Merge dan selection sort 11. Quick sort dan complexita algoritma 12. Struct dan Linked List 13. Antrian dan Stack 14. Graph berarah dan tidak berarah 15. Tree 	

Pustaka	Utama :	<ol style="list-style-type: none"> 1. Munir, Rinaldi dan Lidya, Leony. 2016. Algoritma dan Pemrograman dalam Bahasa Pascal, C dan C++ Edisi Keenam. Informatika : Bandung 2. Gozali, William dan Aji, Alham Fikri. 2014. Pemrograman Kompetitif Dasar. Ikatan Alumni Tim Olimpiade Komputer Indonesia. 3. Zakaria, Teddy M dan Prijono, Agus. 2005. Konsep dan Implementasi Struktur Data. Informatika : Bandung. 4. Niswar, M., Ilham, A., Zainuddin, Z., Adnan, A., P, A., Warni, E., Aswad, I., & Muslimin, Z. (2021). Sosialisasi Metode Berfikir Komputasional pada Pendidikan Dasar dan Menengah di Lingkup Sulawesi Selatan. URNAL EPAT eknologi erapan ntuk engabdian asyarakat, 4(1), 46-52. https://doi.org/10.25042/jurnal_tepat.v4i1.172 						
	Pendukung:							
	Tuliskan pustaka pendukung jika ada, sebagai pengayaan literasi							
Dosen Pengampu		<ol style="list-style-type: none"> 1. Dr. Ir. Ingrid Nurtanio., MT 2. Dr. Amil Ahmad Ilham., ST., M.IT 3. Adnan., ST., MT., PhD 4. Anugrayani Bustamin., ST., MT 5. Elly Warni, S.T., M.T 						
Matakuliah syarat		Dasar Pemrograman Komputer						
				Bentuk Pembelajaran,				
Pekan Ke-	Sub-CPMK (Kemampuan akhir tiap tahapan belajar)	Penilaian		Metode Pembelajaran, Penugasan Mahasiswa, [Estimasi Waktu]	Materi Pembelajaran [Pustaka]		Bobot Penilaian (%)	
		Indikator	Kriteria & Bentuk	Luring (offline)			Daring (online)	
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	

1	Mahasiswa mampu mengetahui konsep dan pengertian Algoritma	Mengetahui konsep definisi, manfaat, mekanisme dan kontribusi Algoritma	Assignments	Lecture TM: 4 x 50 menit.	Lecture VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Konsep dan definisi algoritma 2. Mekanisme dan Kontribusi Algoritma	5%
2	Mahasiswa mampu memahami Dasar Algoritma CPL 3- SubCPMK 2	Memahami Dasar Algoritma (tipe data, aturan penulisan, proses, instruksi dan aksi)	quizzes	Lecture, case study TM: 4 x 50 menit.	Lecture, case study VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Dasar Algoritma (tipe data, aturan penulisan, proses, instruksi, dan aksi) Pustaka : [1]	5%
3	Mahasiswa mampu mengenali Flowchart CPL 4- SubCPMK 3	Mampu mengenali Flowchart (simbol, variabel, algoritma cabang, dan runtunan)	Assignments	Lecture, small group discussion TM: 4 x 50 menit.	Lecture, small group discussion VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit	Flowchart (Simbol, variabel, algoritma cabang, runtunan) Pustaka : [1]	7%

4	Mahasiswa mampu memahami bentuk pemilihan (alih kontrol dan pengulangan)	Mampu memahami bentuk pemilihan (alih kontrol dan pengulangan)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Pemilihan Kontrol Pengulangan) Pustaka : [1]	(Alir dan 7%
5	Mahasiswa mampu mengimplementasikan prosedur, fungsi dan pemrosesan teks	Mampu mengimplementasikan pemanggilan prosedur, pemanggilan fungsi dan pemrosesan teks	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Pemanggilan Prosedur 2. Pemanggilan Fungsi 3. Pemrosesan Teks Pustaka : [1]	7%
6	Mahasiswa mampu mempraktekkan konsep array	Mampu mempraktekkan konsep array seperti menginisialisasi array, mengisi elemen, mencari	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Menginisialisasi array 2. Mengisi elemen array 3. Mencari nilai maksimum dan minimum	7%

		nilai maksimum dan minimum			/104D4224/inde x.php?id_session =13610	Pustaka : [1][2]	
7	Mahasiswa mampu menetapkan ciri dari teknik pencarian	Mampu menetapkan ciri dari teknik pencarian ● Binary Search ● Sequential Search	Assignments quizzes	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Binary Search 2. Sequential Search Pustaka : [2][3]	7%
8	Evaluasi Tengah Semester / Ujian Tengah Semester						
9	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting ● Bubble Sort ● Insertion Sort	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Bubble Sort 2. Insertion Sort Pustaka : [2][3]	9%
	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting Merge Sort	Assignments	Lecture, practicum TM: 4 x 50	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit.	1. Merge Sort Selection Sort Pustaka : [2][3]	9%

10		Selection Sort		menit.	BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610		
11	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting Quick Sort berserta complexitas Algoritmanya	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Quick Sort 2. Complexitas Algoritma Pustaka : [2][3]	9%
12	Mahasiswa mampu memahami struct dan konsep linked list	Mampu memahami struct dan konsep dan operasi dasar linked list (deklarasi simpul, penambahan simpul pada linked list awal, akhir dan tengah)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Konsep Struct dan implementasinya 2. Deklarasi list 3. Penambahan simpul pada list awal, akhir dan tengah Pustaka : [2][3]	7%

13	Mahasiswa mampu memahami Antrian dan Stack	Mampu memahami Antrian dan Stack dalam List	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Antrian dan Stack dalam list Pustaka : [2][3]	7%
14	Mahasiswa mampu memahami teknik graph	mampu memahami teknik graph (graph berarah dan tidak berarah menggunakan linked list)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Graph berarah Graph tidak berarah Pustaka : [2][3]	7%
15	Mahasiswa mampu mengimplementasi konsep Tree	Mampu mengimplementasi konsep Tree menggunakan linked list	Assignments quizzes	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit	Konsep tree menggunakan linked List Pustaka : [2][3]	7%

					https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610		
16	Evaluasi Akhir Semester / Ujian Akhir Semester						



MODUL ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA

MODUL IX

STRUCT DAN LINKED LIST

Pengantar

Modul ini dirancang untuk membantu mahasiswa memahami dasar-dasar struktur data seperti struct dan linked list, serta bagaimana keduanya digunakan dalam pemrograman untuk mengelola data yang lebih kompleks. Dalam rangka mencapai capaian pembelajaran mata kuliah (CPMK) yang telah ditetapkan, mahasiswa akan diarahkan untuk memahami dan mengidentifikasi penerapan struct dan linked list dalam berbagai konteks pemrograman.

Pada Sub-CPMK kali ini, mahasiswa diharapkan mampu mengenali dan memahami cara kerja struct, sebagai tipe data yang memungkinkan pengelompokan berbagai tipe data menjadi satu kesatuan. Selain itu, mahasiswa juga akan mempelajari linked list, struktur data dinamis yang memungkinkan penataan data secara efisien dalam memori. Kegiatan belajar akan dilakukan melalui diskusi kelompok kecil dan pembelajaran melalui ceramah serta didukung oleh platform pembelajaran daring.

Kegiatan pembelajaran ini akan mencakup,

- Tatap Muka (TM): 4 sesi, masing-masing 50 menit.
 - Virtual Classroom (VC): 4 sesi, masing-masing 50 menit.
 - Praktikum Terbimbing (PT): 4 sesi, masing-masing 60 menit.
 - Bimbingan Mandiri (BM): 4 sesi, masing-masing 60 menit
- 

Sumber pembelajaran yang dapat diakses untuk mendukung pemahaman lebih lanjut tersedia di platform sikola, dengan pustaka referensi yang telah disediakan.

Setelah mengikuti modul ini, mahasiswa diharapkan dapat mencapai tingkat pemahaman yang cukup, setidaknya 80%, sebelum melanjutkan ke materi berikutnya

KEGIATAN BELAJAR 1

STRUCT DAN LINKED LIST

A. Deskripsi Singkat

Mahasiswa yang mempelajari struktur data seperti struct dan linked list sering kali menghadapi tantangan dalam memahami konsep dasar serta implementasinya dalam pemrograman. Pemahaman tentang kedua struktur data ini penting karena keduanya memiliki peran yang penting dalam pengelolaan data yang lebih kompleks dan dinamis.

Dalam kegiatan belajar pertama ini, mahasiswa akan diperkenalkan pada konsep dasar struct, sebuah struktur data yang memungkinkan pengelompokan berbagai tipe data yang berbeda dalam satu unit, serta cara penggunaannya untuk menyimpan dan mengelola data secara terorganisir. Selain itu, mahasiswa juga akan mempelajari konsep dasar dan jenis-jenis linked list, struktur data dinamis yang memungkinkan penataan data secara efisien di memori dengan menggunakan node yang saling terhubung.

Teori yang akan disampaikan mencakup berbagai aplikasi praktis dari struct dan linked list, seperti penggunaan struct dalam menyimpan objek dengan atribut yang berbeda-beda, serta pemahaman linked list dalam mengelola data yang sering mengalami perubahan (penambahan atau penghapusan data). Mahasiswa akan diajarkan cara mendeklarasikan, menginisialisasi, dan mengoperasikan struct, serta teknik dasar dalam pemrosesan data yang berbasis pada struct dan linked list.

Dengan pemahaman yang baik, mahasiswa diharapkan dapat memanfaatkan struct dan linked list untuk menyelesaikan berbagai masalah dalam pemrograman dan algoritma yang melibatkan pengelolaan data yang dinamis dan efisien.

B. Relevansi

Materi dalam modul sebelumnya membahas tentang struktur data dasar seperti array dan string, yang penting untuk menyimpan dan mengelola data dalam bentuk yang sederhana dan terstruktur. Pemahaman tentang array dan string sangat berguna saat mahasiswa mulai mempelajari struktur data yang lebih kompleks, seperti struct dan linked list. Keduanya sering digunakan dalam situasi di mana pengelolaan data yang dinamis dan lebih terorganisir diperlukan, seperti dalam aplikasi yang memerlukan pengelompokan data berbeda atau penambahan dan penghapusan elemen secara efisien.

Struktur data struct memungkinkan mahasiswa untuk memodelkan objek yang memiliki banyak atribut yang berbeda dalam satu unit, sedangkan linked list memberikan solusi efisien untuk mengelola data yang sering berubah (penambahan dan penghapusan data). Dalam dunia nyata, struct dan linked list digunakan dalam berbagai aplikasi, seperti sistem basis data, pengelolaan memori, dan pemrograman objek.

Secara keseluruhan, pemahaman struct dan linked list dalam rangkaian modul ini sangat relevan terhadap pencapaian Sub-Capaian Pembelajaran Mata Kuliah (Sub-CPMK). Modul ini melatih kemampuan mahasiswa dalam menggunakan struct maupun linked list untuk data yang lebih kompleks dan pengelolaan data yang lebih canggih. Dengan pemahaman tentang struct dan linked list, mahasiswa diharapkan dapat mengembangkan kemampuan dalam merancang solusi pemrograman yang efisien dan fleksibel, serta dapat lebih mudah menangani tantangan yang melibatkan pengelolaan data dalam aplikasi nyata.

C. Pembelajaran

Capaian pembelajaran yang diharapkan pada kegiatan belajar yang pertama ini adalah peserta kuliah mampu memahami konsep dasar struktur data struct dan linked list, mengimplementasikannya dalam kode C++. Untuk mencapai kompetensi lulusan ini, peserta kuliah diharapkan membaca dan menyelesaikan Kegiatan Belajar 1 ini. Untuk mencapai capaian pembelajaran pada Kegiatan Belajar 1, peserta kuliah harus mengikuti urutan kegiatan pembelajaran yang telah disiapkan dalam modul ini.

1. Uraian

1.1 Struct

Structure atau Struct adalah kumpulan dari beberapa variabel dengan beragam tipe data yang dibungkus dalam satu variabel. Masing-masing elemen data tersebut dikenal dengan sebutan field. Field data tersebut dapat memiliki tipe data yang sama ataupun berbeda. Walaupun field-field tersebut berada dalam satu kesatuan, masing-masing field tersebut tetap dapat diakses secara individual. Kadang-kadang, kita membutuhkan suatu tipe data yang sifatnya komposit; terdiri dari beberapa data lainnya. Contoh kasusnya adalah ketika kita butuh suatu representasi dari titik. Setiap titik pada bidang memiliki dua komponen, yaitu x dan y.

Untuk contoh, misalnya kita ingin menyimpan data barang. Kita bisa saja melakukannya seperti ini:

```
string namaBarang1 = "Pensil"; int hargaBarang1 = 2000; int  
stokBarang1 = 100;
```

Lalu bagaimana jika lebih dari satu barang? Mungkin bisa saja kita buat seperti ini

```
string namaBarang1 = "Pensil"; int hargaBarang1 = 2000; int stokBarang1 = 100;
```

```
string namaBarang2 = "Pulpen"; int hargaBarang2 = 3000; int stokBarang2 = 50; string namaBarang3 = "Buku"; int hargaBarang3 = 5000; int stokBarang3 = 200;
```

Dan bagaimana jika 10 barang? Untuk tidak membuat banyak variabel seperti ini, maka variabel-variabel yang masih dalam satu kelompok bisa kita bungkus di dalam struct.

Struct dapat kita buat dengan kata kunci struct kemudian diikuti dengan nama struct dan isinya.

Struct Barang

```
{  
    string namaBarang;  
    int hargaBarang;  
    int stokBarang;  
};
```

Agar struct dapat digunakan, kita harus membuat variabel dengan tipe data struct. Contoh:

```
Barang brng1;
```

Pada contoh ini, kita membuat variabel **brng1** dengan tipe data Barang yang mana tipe data Barang ini adalah nama struct-nya.

Lalu untuk mengisi nilai ke variabel **brng1**, bisa dilihat di kode C++ berikut ini:

```
#include <iostream>  
#include <string>  
using namespace std;
```

```

struct Barang {
    string namaBarang;
    int hargaBarang;
    int stokBarang;
};

int main() {
    // Cara 1: Inisialisasi menggunakan dot notation
    Barang barang1;
    barang1.namaBarang = "Pensil";
    barang1.hargaBarang = 2000;
    barang1.stokBarang = 100;

    cout << "Nama Barang: " << barang1.namaBarang
        << ", Harga: " << barang1.hargaBarang
        << ", Stok: " << barang1.stokBarang << endl;

    // Cara 2: Designated Initializers (C++20 ke atas)
    Barang barang2 = {
        .namaBarang = "Pulpen",
        .hargaBarang = 3000,
        .stokBarang = 50
    };

    cout << "Nama Barang: " << barang2.namaBarang
        << ", Harga: " << barang2.hargaBarang
        << ", Stok: " << barang2.stokBarang << endl;

    // Cara 3: Aggregate Initialization (C++11 ke atas)
    Barang barang3 = {"Buku", 5000, 200};

    cout << "Nama Barang: " << barang3.namaBarang
        << ", Harga: " << barang3.hargaBarang
        << ", Stok: " << barang3.stokBarang << endl;

    return 0;
}

```

Penjelasan lebih lanjut mengenai inisialisasinya:

- a. **Dot Notation**: Menginisialisasi setiap anggota secara manual setelah deklarasi variabel.
- b. **Designated Initializers**: Memungkinkan kita menentukan nilai dari setiap anggota secara eksplisit saat deklarasi, tapi hanya tersedia di C++20 ke atas.
- c. **Aggregate Initialization**: Tersedia di C++11, menginisialisasi anggota secara berurutan tanpa harus menyebutkan nama anggota.

1.2 Linked List

Linked List adalah jenis struktur data yang berisi kumpulan data yang disusun secara linear dengan setiap data disimpan dalam sebuah simpul dan antara satu simpul dengan simpul lain dihubungkan melalui pointer. Pada linked list tipe data pointer bersifat dinamis, variabel akan dialokasikan hanya pada saat dibutuhkan dan sesudah tidak dibutuhkan dapat direlokasikan kembali. Dengan kata lain, setiap elemen di Linked List "menunjuk" ke elemen berikutnya.

Fitur	Array	Linked List
Ukuran	Tetap	Bisa berubah-ubah
Akses ke Elemen	$O(1)$ - cepat	$O(n)$ – lebih lambat
Penyimpanan di Memori	Berdampingan	Tersebar
Menambah/Menghapus	Lebih sulit (harus geser)	Lebih mudah

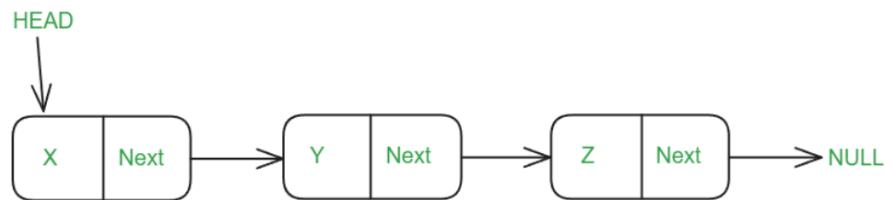
Kita bisa menggunakan Linked List saat kita memerlukan:

- Menyimpan data dengan ukuran yang bisa berubah-ubah.

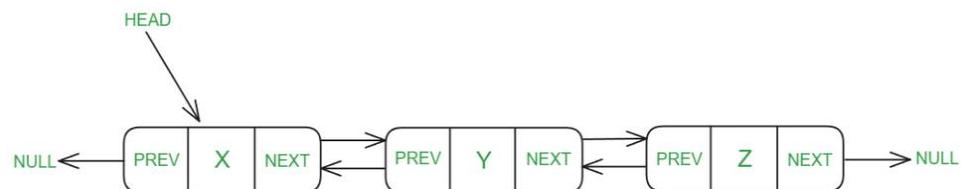
- Menambahkan atau menghapus elemen dengan mudah, terutama di bagian awal atau tengah.

Dan jenis-jenis Linked List berupa:

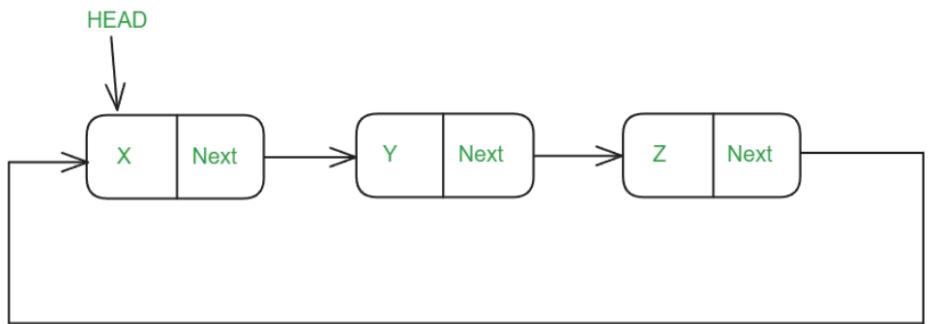
- Singly Linked List (Linked List Sejajar)**: Struktur data di mana setiap node hanya memiliki pointer ke node berikutnya, memungkinkan traversal hanya satu arah, dari awal hingga akhir. Ini sederhana dan hemat memori, tetapi tidak mendukung akses ke elemen sebelumnya.



- Doubly Linked List (Linked List Dua Arah)**: Setiap node memiliki dua pointer, satu menunjuk ke node berikutnya dan satu ke node sebelumnya, sehingga memungkinkan traversal dua arah. Struktur ini memudahkan penambahan dan penghapusan node di kedua arah tetapi memerlukan lebih banyak memori.



- Circular Linked List (Linked List Melingkar)**: Node terakhir menunjuk kembali ke node pertama, membentuk lingkaran. Ini dapat berbentuk singly atau doubly linked, bergantung pada jumlah pointer di setiap node. Struktur ini memungkinkan traversal berkelanjutan dari awal atau akhir tanpa titik berhenti.



2. Rangkuman

Struct adalah kumpulan variabel dengan tipe data berbeda yang disatukan dalam satu variabel komposit untuk menyederhanakan penyimpanan data terkait, seperti data barang atau titik. Dengan struct, kita bisa mengelompokkan data tanpa mendeklarasikan banyak variabel terpisah, membuat kode lebih rapi dan mudah dikelola. Misalnya, “Barang” dapat menyimpan nama, harga, dan stok dalam satu entitas. Sementara itu, Linked List adalah struktur data dinamis yang terdiri dari node yang saling terhubung menggunakan pointer, memungkinkan penambahan dan penghapusan data dengan mudah meskipun aksesnya lebih lambat dibandingkan array. Ada beberapa jenis linked list, yaitu “Singly”, “Doubly”, dan “Circular”, masing-masing memiliki kelebihan dalam cara traversal dan pengelolaan memori sesuai kebutuhan aplikasi.

3. Pustaka

- Dharmayanti, D. (2011). *Algoritma & Pemrograman*.
- Muliono, R. (2019). *Pemrograman C++: Algoritma & Struktur Data*.

D. Penilaian Kegiatan Belajar

1. Penilaian

1.1 Latihan

1) Soal Pertama

Buatlah sebuah program untuk mengelola data buku di perpustakaan. Program ini harus menggunakan struct untuk menyimpan data. Setiap buku memiliki informasi berikut:

Setiap buku memiliki atribut:

- **ID Buku** (integer)
- **Judul Buku** (string)
- **Penulis Buku** (string)

Program ini akan memiliki fungsi untuk:

1. Menambahkan buku ke dalam array.
2. Menampilkan semua buku yang ada di array.

2) Soal Kedua

Sebuah perusahaan ingin menyimpan data karyawannya.

Setiap karyawan memiliki informasi sebagai berikut:

Buatlah program dengan ketentuan sebagai berikut:

1. Definisi Struct Karyawan:

Definisikan struct Karyawan untuk menyimpan data ID, nama, jabatan, dan gaji setiap karyawan.

2. Fungsi untuk Menambahkan Karyawan:

Buat fungsi tambahKaryawan yang menambahkan data karyawan baru ke dalam array atau linked list.

3. Fungsi untuk Menampilkan Daftar Karyawan:

Buat fungsi tampilanKaryawan untuk menampilkan semua data karyawan yang tersimpan.

4. Fungsi untuk Menghitung Total Gaji Karyawan:

Buat fungsi hitungTotalGaji untuk menghitung dan mengembalikan total gaji seluruh karyawan di perusahaan.

5. Fungsi untuk Mencari Karyawan berdasarkan ID:

Buat fungsi cariKaryawan yang menerima ID karyawan sebagai parameter dan menampilkan informasi karyawan tersebut jika ditemukan. Jika tidak ditemukan, tampilkan pesan "Karyawan tidak ditemukan".

2. Umpam Balik dan Tindak Lanjut

1) Source Code Soal Pertama

```
#include <iostream>
#include <string>
using namespace std;

// Definisi struct untuk Buku
struct Buku {
    int id;
    string judul;
    string penulis;
};

// Fungsi untuk menambahkan buku ke dalam array
void tambahBuku(Buku koleksi[], int& jumlahBuku, int id, string judul, string penulis) {
    koleksi[jumlahBuku].id = id;
    koleksi[jumlahBuku].judul = judul;
    koleksi[jumlahBuku].penulis = penulis;
    jumlahBuku++;
}

// Fungsi untuk menampilkan semua buku dalam array
void tampilanBuku(const Buku koleksi[], int jumlahBuku) {
    for (int i = 0; i < jumlahBuku; i++) {
        cout << "ID Buku: " << koleksi[i].id <<
    endl;
```

```

        cout << "Judul Buku: " << koleksi[i].judul
<< endl;
        cout << "Penulis Buku: " <<
koleksi[i].penulis << endl;
        cout << "-----" << endl;
    }
}

int main() {
    Buku koleksi[100]; // Array statis untuk
menyimpan buku
    int jumlahBuku = 0; // Jumlah buku yang ada
dalam array

    // Menambahkan beberapa buku ke dalam array
    tambahBuku(koleksi, jumlahBuku, 1, "Pemrograman
C++", "Budi Santoso");
    tambahBuku(koleksi, jumlahBuku, 2, "Algoritma
dan Struktur Data", "Siti Aminah");
    tambahBuku(koleksi, jumlahBuku, 3, "Jaringan
Komputer", "Marcellino");

    // Menampilkan semua buku dalam array
    cout << "Daftar Buku di Perpustakaan:" << endl;
    tampilanBuku(koleksi, jumlahBuku);

    return 0;
}

```

2) Source Code Soal Kedua

```

#include <iostream>
#include <string>
using namespace std;

// Definisi struct untuk Karyawan
struct Karyawan {
    int id;
    string nama;

```

```

        string jabatan;
        float gaji;
    };

// Fungsi untuk menambahkan karyawan ke dalam array
void tambahKaryawan(Karyawan karyawan[], int
&jumlahKaryawan) {
    cout << "Masukkan ID Karyawan: ";
    cin >> karyawan[jumlahKaryawan].id;
    cin.ignore(); // Mengabaikan newline

    cout << "Masukkan Nama Karyawan: ";
    getline(cin, karyawan[jumlahKaryawan].nama);

    cout << "Masukkan Jabatan: ";
    getline(cin, karyawan[jumlahKaryawan].jabatan);

    cout << "Masukkan Gaji: ";
    cin >> karyawan[jumlahKaryawan].gaji;

    jumlahKaryawan++;
    cout << "Karyawan berhasil ditambahkan!" <<
endl;
}

// Fungsi untuk menampilkan semua karyawan
void tampilanKaryawan(const Karyawan karyawan[],
int jumlahKaryawan) {
    if (jumlahKaryawan == 0) {
        cout << "Tidak ada karyawan." << endl;
    } else {
        cout << "Daftar Karyawan:" << endl;
        for (int i = 0; i < jumlahKaryawan; i++) {
            cout << "ID: " << karyawan[i].id
                << ", Nama: " << karyawan[i].nama
                << ", Jabatan: " <<
karyawan[i].jabatan
                << ", Gaji: " << karyawan[i].gaji
            << endl;
    }
}

```

```

        cout << "-----"
<< endl;
    }
}
}

// Fungsi untuk menghitung total gaji karyawan
float hitungTotalGaji(const Karyawan karyawan[], int jumlahKaryawan) {
    float totalGaji = 0;
    for (int i = 0; i < jumlahKaryawan; i++) {
        totalGaji += karyawan[i].gaji;
    }
    return totalGaji;
}

// Fungsi untuk mencari karyawan berdasarkan ID
void cariKaryawan(const Karyawan karyawan[], int jumlahKaryawan, int id) {
    bool ditemukan = false;
    for (int i = 0; i < jumlahKaryawan; i++) {
        if (karyawan[i].id == id) {
            cout << "Karyawan ditemukan:" << endl;
            cout << "ID: " << karyawan[i].id
                << ", Nama: " << karyawan[i].nama
                << ", Jabatan: " <<
karyawan[i].jabatan
                << ", Gaji: " << karyawan[i].gaji
<< endl;
            ditemukan = true;
            break;
        }
    }
    if (!ditemukan) {
        cout << "Karyawan dengan ID " << id << "
tidak ditemukan." << endl;
    }
}

```

```

int main() {
    Karyawan karyawan[100]; // Array untuk menyimpan
    data karyawan
    int jumlahKaryawan = 0; // Jumlah karyawan yang
    ada
    int pilihan;

    do {
        cout << "\nMenu:" << endl;
        cout << "1. Tambah Karyawan" << endl;
        cout << "2. Tampilkan Semua Karyawan" <<
        endl;
        cout << "3. Hitung Total Gaji" << endl;
        cout << "4. Cari Karyawan berdasarkan ID" <<
        endl;
        cout << "5. Keluar" << endl;
        cout << "Pilih opsi: ";
        cin >> pilihan;

        if (pilihan == 1) {
            tambahKaryawan(karyawan,
jumlahKaryawan);
        } else if (pilihan == 2) {
            tampilkanKaryawan(karyawan,
jumlahKaryawan);
        } else if (pilihan == 3) {
            float totalGaji =
hitungTotalGaji(karyawan, jumlahKaryawan);
            cout << "Total Gaji Karyawan: " <<
totalGaji << endl;
        } else if (pilihan == 4) {
            int id;
            cout << "Masukkan ID Karyawan yang ingin
dicari: ";
            cin >> id;
            cariKaryawan(karyawan, jumlahKaryawan,
id);
        }
    } while (pilihan != 5);
}

```

```
    cout << "Program selesai." << endl;
    return 0;
}
```



ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA

MODUL X

Antrian dan Stack



Elly Warni
Ingrid Nurtanio
Amil Ahmad Ilham
Adnan
Anugrayani Bustamin

PRAKATA

Alhamdulillah, puji dan syukur kami panjatkan ke hadirat Allah SWT yang telah melimpahkan rahmat dan hidayah-Nya, sehingga modul ini dapat terselesaikan dengan baik. Shalawat serta salam senantiasa tercurah kepada junjungan kita, Nabi Muhammad SAW, beserta keluarga dan sahabat beliau yang telah membawa kita dari zaman kegelapan menuju terang benderang ilmu pengetahuan.

Modul ini disusun sebagai bagian dari rangkaian pembelajaran yang bertujuan untuk membantu mahasiswa memahami dan menerapkan konsep dasar struktur data stack dan queue, serta memanfaatkan kedua struktur data ini untuk menyelesaikan permasalahan. Modul ini merupakan lanjutan dari modul sebelumnya yang membahas dasar-dasar algoritma, sehingga diharapkan mahasiswa memiliki pemahaman yang lebih mendalam dan terstruktur.

Tidak dapat dipungkiri bahwa proses penyusunan modul ini tidak lepas dari dukungan berbagai pihak. Oleh karena itu, kami mengucapkan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan masukan dan bimbingan dalam proses penyusunan modul ini. Terima kasih kami sampaikan kepada rekan-rekan dosen dan staf yang telah memberikan kritik dan saran yang membangun, sehingga modul ini dapat terselesaikan dengan baik.

Kami menyadari bahwa modul ini masih memiliki keterbatasan dan kekurangan. Oleh karena itu, kami sangat mengharapkan masukan, kritik, dan saran dari para pembaca untuk penyempurnaan di masa yang akan datang. Semoga modul ini dapat memberikan manfaat dan menjadi bahan



referensi yang berharga bagi para mahasiswa dalam memahami konsep algoritma dan pemrograman.

Akhir kata, semoga upaya ini mendapat ridha Allah SWT dan memberikan kontribusi positif bagi kemajuan ilmu pengetahuan. Amin.

Makassar, November 2024

Penulis

DAFTAR ISI

PRAKATA	i
DAFTAR ISI	iii
RPS MATA KULIAH ALGORITMA DAN STRUKTUR DATA	iv
MODUL ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA	13
Pengantar	13
KEGIATAN BELAJAR 1	15
A. Deskripsi Singkat	15
B. Relevansi	16
C. Pembelajaran	17
D. Penilaian Kegiatan Belajar	22

RPS MATA KULIAH ALGORITMA DAN STRUKTUR DATA



**UNIVERSITAS HASANUDDIN
FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK INFORMATIKA**

Kode Dokumen

RENCANA PEMBELAJARAN SEMESTER

MATA KULIAH (MK)	KODE	Rumpun MK	BOBOT (sks)		SEMESTER	Tgl Penyusunan			
Algoritma Pemrograman dan Struktur Data	23D12110604	Teknik Informatika	T=2	P=2	2	19-02-2023			
OTORISASI	Pengembang RPS Dr. Ir. Ingrid Nurtanio., MT Dr. Amil Ahmad Ilham., ST., M.IT Adnan., ST., MT., PhD Anugrayani Bustamin., ST., MT Elly Warni, S.T., M.T	Koordinator RMK Dr. Ir. Ingrid Nurtanio., MT	Ketua PRODI Prof. Dr. Indrabayu., ST., M.T., M.Bus.Sys						
Capaian Pembelajaran (CP)	CPL-PRODI yang dibebankan pada MK <i>Intended Learning Outcomes</i>								
	CPL 1	Memiliki dasar pengetahuan Teknik Informatika yang meliputi teori dan konsep dasar dari Ilmu Komputer, Matematika dan Statistika, Algoritma dan Pemrograman, Rekayasa Perangkat Lunak, Manajemen Informasi dan Ketahanan Digital, serta pengetahuan tingkat lanjut pada bidang-bidang khusus Teknik Informatika, seperti Kecerdasan Buatan, Data Science, Jaringan Komputer, Komputasi Awan dan Internet of Things.							
	CPL 3	Mampu mengaplikasikan pengetahuan bidang Teknik Informatika yang dipadankan dengan bidang ilmu lainnya untuk menganalisa dan mencari solusi dari berbagai masalah berbasis komputasi.							
	CPL 4	Mampu mendesain, mengimplementasikan dan mengevaluasi solusi berbasis komputasi dengan mengaplikasikan ilmu Teknik Informatika dan dasar-dasar pembangunan perangkat lunak.							
	Capaian Pembelajaran Mata Kuliah (CPMK)								
	CPMK	Setelah mengikuti Mata Kuliah Algoritma Pemrograman dan Struktur Data selama 1 (satu) semester, mahasiswa mampu menguasai dan menerapkan konsep teoritis bidang Informatika dan ilmu komputer secara dan mendalam serta memformulasikan penyelesaian masalah procedural dengan pemikiran logis dan sistematis secara mandiri dan terukur sebagai dasar pengembangan perangkat lunak.							
CPL ⇒ Sub-CPMK									
	CPL 1	Sub-CPMK 1 : Mahasiswa mampu mendefinisikan konsep dan pengertian Algoritma							

	CPL 1	Sub-CPMK 2 : Mahasiswa mampu mendeksripsikan Dasar Algoritma
	CPL 3	Sub-CPMK 3 : Mahasiswa mampu mengimplementasikan Flowchart
	CPL 1	Sub-CPMK 4 : Mahasiswa mampu memahami bentuk pemilihan dan perulangan
	CPL 4	Sub-CPMK 5 : Mahasiswa mampu mengimplementasikan prosedur, fungsi dan pemrosesan teks
	CPL 1	Sub-CPMK 6 : Mahasiswa mampu mempraktekkan konsep array
	CPL 4	Sub-CPMK 7 : Mahasiswa mampu menetapkan ciri dari teknik pencarian
	CPL 4	Sub-CPMK 8 : Mahasiswa mampu menetapkan ciri dari teknik sorting dan menganalisis complexitas algoritma
	CPL 1	Sub-CPMK 9 : Mahasiswa mampu memahami struct dan konsep linked list
	CPL 1	Sub-CPMK 10 : Mahasiswa mampu memahami Antrian dan Stack
	CPL 1	Sub-CPMK 11 : Mahasiswa mampu memahami teknik graph
	CPL 3	Sub-CPMK 12 : Mahasiswa mampu mengimplementasi konsep Tree
Deskripsi Singkat MK	<p>Mata kuliah Algoritma Pemrograman dan Struktur Data merupakan mata kuliah yang menyajikan konsep pemrograman secara mendalam terkait algoritma dan struktur data. Muatan mata kuliah merupakan dasar dari ilmu matematika, algoritma dan logika yang menjadi syarat utama pada mata kuliah Pemrograman lanjut . Mata kuliah ini disajikan untuk mahasiswa semester dua di Program Studi Teknik Informatika, Penanggungjawab mata kuliah adalah Laboratorium Komputer . Metode Pembelajaran bauran, teori dan praktik</p>	
Bahan Kajian / Materi Pembelajaran	<ol style="list-style-type: none"> 1. Konsep Algoritma 2. Dasar Algoritma (Aturan Penulisan Header, deklarasi dan deskripsi) 3. Flowchart (Simbol, variabel, algoritma cabang, runtunan) 4. Pemilihan (Alir Kontrol dan Pengulangan) 5. Prosedur, Fungsi dan Pemrosesan Teks 6. Array – Larik 7. Pencarian (Binary Search) 8. Pencarian (Sequential Search) 9. Bubble dan Insertion Sort 10. Merge dan selection sort 11. Quick sort dan complexita algoritma 12. Struct dan Linked List 13. Antrian dan Stack 14. Graph berarah dan tidak berarah 15. Tree 	

Pustaka	Utama :	<ol style="list-style-type: none"> 1. Munir, Rinaldi dan Lidya, Leony. 2016. Algoritma dan Pemrograman dalam Bahasa Pascal, C dan C++ Edisi Keenam. Informatika : Bandung 2. Gozali, William dan Aji, Alham Fikri. 2014. Pemrograman Kompetitif Dasar. Ikatan Alumni Tim Olimpiade Komputer Indonesia. 3. Zakaria, Teddy M dan Priyono, Agus. 2005. Konsep dan Implementasi Struktur Data. Informatika : Bandung. 4. Niswar, M., Ilham, A., Zainuddin, Z., Adnan, A., P, A., Warni, E., Aswad, I., & Muslimin, Z. (2021). Sosialisasi Metode Berfikir Komputasional pada Pendidikan Dasar dan Menengah di Lingkup Sulawesi Selatan. URNAL EPAT eknologi erapan ntuk engabdian asyarakat, 4(1), 46-52. https://doi.org/10.25042/jurnal_tepat.v4i1.172 					
	Pendukung:						
	Tuliskan pustaka pendukung jika ada, sebagai pengayaan literasi						
Dosen Pengampu		<ol style="list-style-type: none"> 1. Dr. Ir. Ingrid Nurtanio., MT 2. Dr. Amil Ahmad Ilham., ST., M.IT 3. Adnan., ST., MT., PhD 4. Anugrayani Bustamin., ST., MT 5. Elly Warni, S.T., M.T 					
Matakuliah syarat		Dasar Pemrograman Komputer					
				Bentuk Pembelajaran,			
Pekan Ke-	Sub-CPMK (Kemampuan akhir tiap tahapan belajar)	Penilaian		Metode Pembelajaran, Penugasan Mahasiswa, [Estimasi Waktu]		Materi Pembelajaran [Pustaka]	Bobot Penilaian (%)
		Indikator	Kriteria & Bentuk	Luring (offline)	Daring (online)		
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)

1	Mahasiswa mampu mengetahui konsep dan pengertian Algoritma	Mengetahui konsep definisi, manfaat, mekanisme dan kontribusi Algoritma	Assignments	Lecture TM: 4 x 50 menit.	Lecture VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Konsep dan definisi algoritma 2. Mekanisme dan Kontribusi Algoritma	5%
2	Mahasiswa mampu memahami Dasar Algoritma CPL 3- SubCPMK 2	Memahami Dasar Algoritma (tipe data, aturan penulisan, proses, instruksi dan aksi)	quizzes	Lecture, case study TM: 4 x 50 menit.	Lecture, case study VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Dasar Algoritma (tipe data, aturan penulisan, proses, instruksi, dan aksi) Pustaka : [1]	5%
3	Mahasiswa mampu mengenali Flowchart CPL 4- SubCPMK 3	Mampu mengenali Flowchart (simbol, variabel, algoritma cabang, dan runtunan)	Assignments	Lecture, small group discussion TM: 4 x 50 menit.	Lecture, small group discussion VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit	Flowchart (Simbol, variabel, algoritma cabang, runtunan) Pustaka : [1]	7%

4	Mahasiswa mampu memahami bentuk pemilihan (alih kontrol dan pengulangan)	Mampu memahami bentuk pemilihan (alih kontrol dan pengulangan)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Pemilihan Kontrol Pengulangan) Pustaka : [1]	(Alir dan 7%
5	Mahasiswa mampu mengimplementasikan prosedur, fungsi dan pemrosesan teks	Mampu mengimplementasikan pemanggilan prosedur, pemanggilan fungsi dan pemrosesan teks	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Pemanggilan Prosedur 2. Pemanggilan Fungsi 3. Pemrosesan Teks Pustaka : [1]	7%
6	Mahasiswa mampu mempraktekkan konsep array	Mampu mempraktekkan konsep array seperti menginisialisasi array, mengisi elemen, mencari	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit	1. Menginisialisasi array 2. Mengisi elemen array 3. Mencari nilai maksimum dan minimum	7%

		nilai maksimum dan minimum			https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Pustaka : [1][2]	
7	Mahasiswa mampu menetapkan ciri dari teknik pencarian	Mampu menetapkan ciri dari teknik pencarian ● Binary Search ● Sequential Search	Assignments quizzes	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Binary Search 2. Sequential Search Pustaka : [2][3]	7%
8	Evaluasi Tengah Semester / Ujian Tengah Semester						
9	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting ● Bubble Sort ● Insertion Sort	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Bubble Sort 2. Insertion Sort Pustaka : [2][3]	9%

10	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting Merge Sort Selection Sort	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Merge Sort Selection Sort Pustaka : [2][3]	9%
11	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting Quick Sort berserta complexitas Algoritmanya	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Quick Sort 2. Complexitas Algoritma Pustaka : [2][3]	9%
12	Mahasiswa mampu memahami struct dan konsep linked list	Mampu memahami struct dan konsep dan operasi dasar linked list (deklarasi simpul, penambahan)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit	1. Konsep Struct dan implementasinya 2. Deklarasi list 3. Penambahan simpul pada list awal, akhir dan tengah	7%

		simpul pada linked list awal, akhir dan tengah)			https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Pustaka : [2][3]	
13	Mahasiswa mampu memahami Antrian dan Stack	Mampu memahami Antrian dan Stack dalam List	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Antrian dan Stack dalam list Pustaka : [2][3]	7%
14	Mahasiswa mampu memahami teknik graph	mampu memahami teknik graph (graph berarah dan tidak berarah menggunakan linked list)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Graph berarah Graph tidak berarah Pustaka : [2][3]	7%



MODUL ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA

MODUL X

ANTRIAN DAN STACK

• Pengantar

Modul ini dirancang untuk membantu mahasiswa memahami struktur data *stack* dan *queue*, termasuk konsep dasar, operasi utama, dan penerapannya. Dalam rangka mencapai capaian pembelajaran mata kuliah (CPMK) yang telah ditetapkan, mahasiswa akan diarahkan untuk mengenali, memahami, dan mengimplementasikan *stack* dan *queue* dalam konteks pemecahan masalah yang terstruktur.

Pada Sub-CPMK kali ini, mahasiswa diharapkan mampu mengenali dan memahami fungsi serta cara kerja dan penggunaan operasi pada *stack* dan *queue*. Kegiatan belajar akan dilakukan melalui diskusi kelompok kecil dan pembelajaran melalui ceramah serta didukung oleh platform pembelajaran daring.

Kegiatan pembelajaran ini akan mencakup,

- Tatap Muka (TM): 4 sesi, masing-masing 50 menit.
- Virtual Classroom (VC): 4 sesi, masing-masing 50 menit.
- Praktikum Terbimbing (PT): 4 sesi, masing-masing 60 menit.
- Bimbingan Mandiri (BM): 4 sesi, masing-masing 60 menit.

Sumber pembelajaran yang dapat diakses untuk mendukung pemahaman lebih lanjut tersedia di platform sekolah, dengan pustaka referensi yang telah disediakan





Setelah mengikuti modul ini, mahasiswa diharapkan dapat mencapai tingkat pemahaman yang cukup, setidaknya 80%, sebelum melanjutkan ke materi berikutnya

KEGIATAN BELAJAR 1

ANTRIAN DAN STACK

A. Deskripsi Singkat

Mahasiswa yang mempelajari *stack* dan *queue* sering kali menemui tantangan dalam memahami konsep serta operasi dasar pada kedua struktur data ini. Pemahaman *stack* dan *queue* sangat penting, karena keduanya merupakan struktur data fundamental yang sering diterapkan dalam berbagai aplikasi pemrograman, seperti pengelolaan data sementara, penjadwalan proses, dan sistem antrian.

Pada kegiatan belajar pertama ini, mahasiswa akan diperkenalkan dengan konsep dasar *stack* dan *queue*, mulai dari definisi, konsep, hingga operasi dasar pada *Stack* seperti *push*, *pop*, *top*, *isEmpty*, dan sebagainya serta operasi dasar pada *Queue* seperti *push*, *pop*, *front*, *isEmpty*, dan sebagainya. *Stack* bekerja dengan prinsip LIFO (*Last In, First Out*), sementara *queue* menggunakan prinsip FIFO (*First In, First Out*), yang membedakan keduanya dalam cara data dikelola dan diakses.

Melalui pemahaman yang tepat terhadap operasi-operasi ini, mahasiswa diharapkan mampu mengembangkan kemampuan dalam menggunakan *stack* dan *queue* untuk menyelesaikan masalah pemrograman yang membutuhkan pengelolaan data secara terstruktur. Pembelajaran juga mencakup penerapan *stack* dan *queue* dalam berbagai kasus praktis yang ada.

Dengan pemahaman yang tepat, mahasiswa diharapkan dapat mengimplementasikan *stack* dan *queue* secara efektif dalam kode pemrograman, serta mengaplikasikan struktur data ini sebagai alat bantu yang penting dalam perancangan solusi pemrograman yang efisien dan terstruktur.

B. Relevansi

Materi dalam modul array sebelumnya memberikan dasar penting dalam pemahaman struktur data, yang menjadi pondasi dalam merancang logika pemrograman. Pada modul tersebut, mahasiswa mempelajari konsep dasar array serta cara mengakses, menyimpan, dan memanipulasi data secara efisien. Pemahaman ini tidak berdiri sendiri, melainkan didukung dengan penguasaan elemen-elemen dasar array yang akan menjadi bekal dalam memahami *stack* dan *queue*. Dengan landasan ini, mahasiswa dapat lebih mudah memahami bagaimana data disusun dan diakses secara sistematis dalam struktur *stack* dan *queue*, yang memiliki pola akses berbeda namun saling melengkapi dalam pengelolaan data dan penyelesaian masalah dalam pemrograman.

Modul ini juga memberikan keterampilan praktis dalam mengimplementasikan *stack* dan *queue* sebagai dasar bagi algoritma yang lebih kompleks di modul-modul selanjutnya. Misalnya, *stack* sering dimanfaatkan dalam pengelolaan fungsi rekursif, terutama pada algoritma pencarian *Depth-First Search* (DFS) pada graf, sedangkan *queue* efektif dalam mengatur data yang memerlukan antrian, seperti pada algoritma pencarian *Breadth-First Search* (BFS) di graf. Pemahaman ini akan membantu mahasiswa dalam merancang algoritma yang lebih efektif dan efisien.

Dalam konteks pencapaian Sub-Capaian Pembelajaran Mata Kuliah (Sub-CPMK), materi *stack* dan *queue* relevan untuk mengembangkan kemampuan mahasiswa dalam memecahkan masalah pemrograman dengan pendekatan yang sistematis. Selain itu, penguasaan *stack* dan *queue* mempersiapkan mahasiswa untuk mempelajari struktur data yang lebih kompleks dan memahami pengelolaan data dalam pemrograman. Mahasiswa diharapkan dapat menggunakan *stack* dan *queue* sebagai alat bantu dalam perancangan algoritma yang terstruktur dan diimplementasikan dengan tepat dalam kode pemrograman.

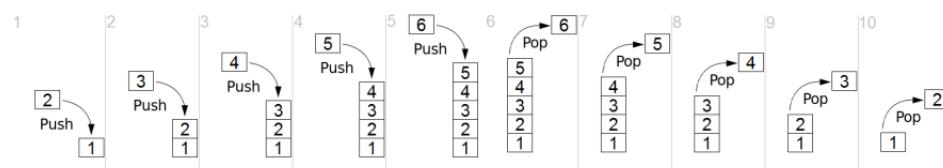
C. Pembelajaran

Capaian pembelajaran yang diharapkan pada kegiatan belajar yang pertama ini adalah peserta kuliah mampu memahami dan menerapkan konsep dasar struktur data stack dan queue, serta memanfaatkan kedua struktur data ini untuk menyelesaikan permasalahan yang memerlukan akses data dengan pola tertentu (LIFO untuk stack dan FIFO untuk queue). Untuk mencapai kompetensi lulusan ini, peserta kuliah diharapkan membaca dan menyelesaikan Kegiatan Belajar 1 ini. Untuk mencapai capaian pembelajaran pada Kegiatan Belajar 1, peserta kuliah harus mengikuti urutan kegiatan pembelajaran yang telah disiapkan dalam modul ini.

1. Uraian

1.1 Stack

Stack dapat dimisalkan seperti tumpukan piring pada umumnya. Jika terdapat piring baru yang ingin dimasukkan, maka piring tersebut masuk dari paling atas. Jika sebuah piring akan diambil dari tumpukan, maka yang diambil juga piring yang paling atas. Struktur data *Stack* menyimpan informasi dalam bentuk seperti tumpukan. Informasi terbaru akan dimasukkan ke paling atas tumpukan. Hanya informasi paling atas yang bisa diakses/dihapus pada setiap waktunya. Oleh karena itu struktur data *Stack* disebut memiliki sifat LIFO (*Last In First Out*).



Stack memiliki operasi sebagai berikut:

- *push*, yaitu memasukkan elemen baru ke bagian atas tumpukan.
- *pop*, yaitu membuang elemen paling atas tumpukan.
- *top*, yaitu mengakses elemen paling atas tumpukan.
- *isEmpty*, yaitu memeriksa apakah tumpukan saat ini kosong.

Anda dapat mengimplementasikan *Stack* menggunakan sebuah array dan variabel penunjuk. Variabel penunjuk ini menyatakan indeks array yang menjadi elemen paling atas *Stack*, dan bergerak maju/mundur sesuai dengan perintah *push/pop*. Seluruh operasi dapat dilakukan dalam O(1).

Contoh kode C++:

```
#include <iostream>
#include <stack>
using namespace std;

int main() {
    stack<int> stack;
    stack.push(21);
    stack.push(22);
    stack.push(24);
    stack.push(25);

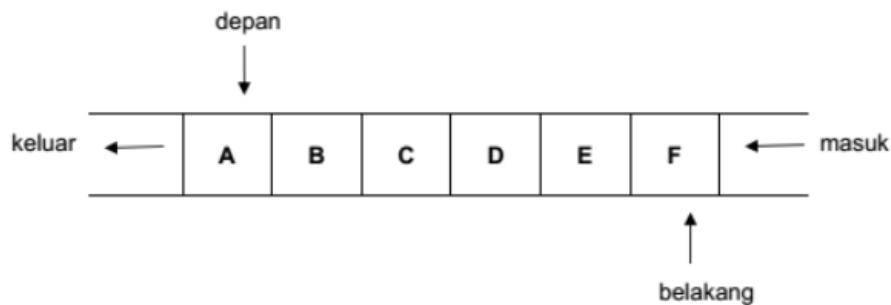
    int num=0;
    stack.push(num);
    stack.pop();
    stack.pop();
    stack.pop();

    while (!stack.empty()) {
        cout << stack.top() << " ";
    }
}
```

```
        stack.pop();  
    }  
}
```

1.2 Queue

Apakah anda pernah melihat antrian pembelian? Struktur data *Queue* mirip dengan analogi antrian tersebut. Saat seorang ingin masuk ke antrian, maka orang tersebut harus mengantri dari belakang. Sementara itu, orang yang dilayani terlebih dahulu adalah orang yang paling depan. Struktur data *Queue* menyimpan informasi dalam bentuk antrian. Informasi yang baru dimasukkan ke paling belakang antrian. Hanya informasi paling depan yang bisa diakses/dihapus pada setiap waktunya. Oleh karena itu struktur data *Queue* disebut memiliki sifat FIFO (*First In First Out*).



Queue memiliki beberapa operasi yang dapat dilakukan:

- *push*, yaitu memasukkan elemen baru ke bagian akhir antrian.
- *pop*, yaitu mengeluarkan elemen paling depan antrian.
- *front*, yaitu mengakses elemen yang paling depan antrian.
- *isEmpty*, yaitu memeriksa apakah antrian saat ini kosong.

Anda dapat mengimplementasikan *Queue* menggunakan sebuah array dan dua variabel penunjuk. Variabel penunjuk ini menyatakan indeks array yang menjadi elemen paling depan

dan belakang Queue. Kedua variabel penunjuk ini selalu bergerak maju. Seluruh operasi dapat dilakukan dalam O(1).

Contoh kode C++:

```
#include <iostream>
#include <queue>

using namespace std;

// Print the queue
void showq(queue<int> gq)
{
    queue<int> g = gq;
    while (!g.empty()) {
        cout << g.front() << " ";
        g.pop();
    }
    cout << '\n';
}

// Driver Code
int main()
{
    queue<int> q;
    q.push(10);
    q.push(20);
    q.push(30);

    cout << "The queue q is : ";
    showq(q);

    cout << "\ngquiz.size() : " << q.size();
    cout << "\ngquiz.front() : " << q.front();
    cout << "\ngquiz.back() : " << q.back();

    cout << "\ngquiz.pop() : ";
    q.pop();
    showq(q);

    return 0;
}
```

2. Rangkuman

Stack adalah struktur data dengan sifat LIFO (*Last In, First Out*), seperti tumpukan piring di mana elemen baru selalu ditambahkan dan diambil dari bagian atas. Operasi dasar *Stack* meliputi *push* (menambahkan elemen), *pop* (menghapus elemen paling atas), *top* (mengakses elemen paling atas tanpa menghapusnya), dan *isEmpty* (memeriksa apakah kosong). Semua operasi ini dapat dilakukan dalam waktu konstan, O(1), menggunakan array dan penunjuk ke elemen teratas

Sebaliknya, *Queue* memiliki sifat FIFO (*First In, First Out*), mirip antrian, di mana elemen baru masuk dari belakang dan yang pertama keluar dari depan. Operasi pada *Queue* mencakup *push* (menambahkan ke belakang), *pop* (menghapus dari depan), *front* (mengakses elemen depan), dan *isEmpty*. *Queue* juga diimplementasikan dengan array dan dua penunjuk untuk bagian depan dan belakang, dan seluruh operasinya juga memiliki waktu konstan O(1).

3. Pustaka.

- a. Aji, Alham Fikri dan Gozali, William. *Pemrograman Kompetitif Dasar*. ISBN 978-602-6598-89-9.
- b. Muliono, R. (2019). *Pemrograman C++: Algoritma & Struktur Data*.

D. Penilaian Kegiatan Belajar

1. Penilaian

1.1 Latihan

1) Soal Mudah (*Stack*)

Format Masukan

Baris pertama berisi sebuah bilangan bulat N.

Contoh Masukan

9

Operasi PUSH

Item paling atas: Piring-1 Jumlah item di Stack: 1

Item paling atas: Piring-2 Jumlah item di Stack: 2

Item paling atas: Piring-3 Jumlah item di Stack: 3

Item paling atas: Piring-4 Jumlah item di Stack: 4

Item paling atas: Piring-5 Jumlah item di Stack: 5

Item paling atas: Piring-6 Jumlah item di Stack: 6

Item paling atas: Piring-7 Jumlah item di Stack: 7

Item paling atas: Piring-8 Jumlah item di Stack: 8

Item paling atas: Piring-9 Jumlah item di Stack: 9

Operasi POP

Item paling atas: Piring-9

Item yang di keluarkan: Piring-9 Sisa item di Stack: 8 Item

paling atas: Piring-8

Item yang di keluarkan: Piring-8 Sisa item di Stack: 7 Item

paling atas: Piring-7

Item yang di keluarkan: Piring-7 Sisa item di Stack: 6 Item

paling atas: Piring-6

Item yang di keluarkan: Piring-6 Sisa item di Stack: 5 Item

paling atas: Piring-5

Item yang di keluarkan: Piring-5 Sisa item di Stack: 4 Item paling atas: Piring-4

Item yang di keluarkan: Piring-4 Sisa item di Stack: 3 Item paling atas: Piring-3

Item yang di keluarkan: Piring-3 Sisa item di Stack: 2 Item paling atas: Piring-2

Item yang di keluarkan: Piring-2 Sisa item di Stack: 1 Item paling atas: Piring-1

Item yang di keluarkan: Piring-1 Sisa item di Stack: 0

2) Soal Mudah (*Queue*)

Format Masukan

Baris pertama berisi sebuah bilangan bulat N.

Contoh Masukan

9

Operasi ENQUEUE

Item paling depan: Piring-1 Jumlah item di Queue: 1

Item paling depan: Piring-1 Jumlah item di Queue: 2

Item paling depan: Piring-1 Jumlah item di Queue: 3

Item paling depan: Piring-1 Jumlah item di Queue: 4

Item paling depan: Piring-1 Jumlah item di Queue: 5

Item paling depan: Piring-1 Jumlah item di Queue: 6

Item paling depan: Piring-1 Jumlah item di Queue: 7

Item paling depan: Piring-1 Jumlah item di Queue: 8

Item paling depan: Piring-1 Jumlah item di Queue: 9

Operasi DEQUEUE

Item paling depan: Piring-1

Item yang di keluarkan: Piring-1 Sisa item di Queue: 8 Item paling depan: Piring-2

Item yang di keluarkan: Piring-2 Sisa item di Queue: 7 Item paling depan: Piring-3

Item yang di keluarkan: Piring-3 Sisa item di Queue: 6 Item paling depan: Piring-4

Item yang di keluarkan: Piring-4 Sisa item di Queue: 5 Item paling depan: Piring-5

Item yang di keluarkan: Piring-5 Sisa item di Queue: 4 Item paling depan: Piring-6

Item yang di keluarkan: Piring-6 Sisa item di Queue: 3 Item paling depan: Piring-7

Item yang di keluarkan: Piring-7 Sisa item di Queue: 2 Item paling depan: Piring-8

Item yang di keluarkan: Piring-8 Sisa item di Queue: 1 Item paling depan: Piring-9

Item yang di keluarkan: Piring-9 Sisa item di Queue: 0

1.2 Tantangan

(<https://tlx.toki.id/courses/competitive-1/chapters/08/problems/C>)

Deskripsi

Pak Dengklek meminta Anda untuk menyimulasikan QQ operasi terhadap sebuah tumpukan (*stack*) yang terdiri dari bilangan bulat.

Pada mulanya, tumpukan tersebut kosong. Lalu, Anda perlu menjalankan serangkaian perintah, yang masing-masing dapat berupa:

- add X Y

masukkan (*push*) YY buah bilangan bulat, masing-masing adalah XX, ke atas tumpukan, lalu cetak banyaknya bilangan pada tumpukan.

- del Y

keluarkan (*pop*) YY buah bilangan dari atas tumpukan, lalu cetak bilangan pertama yang dibuang.

- *adx D*

tambahkan DD pada setiap bilangan pada tumpukan.

- *dex D*

kurangkan DD dari setiap bilangan pada tumpukan.

Format Masukan

Baris pertama berisi sebuah bilangan bulat QQ baris berikutnya masing-masing berisi sebuah perintah sesuai penjelasan di atas.

Format Keluaran

Beberapa baris, sesuai dengan banyaknya perintah add/del yang diberikan pada masukan. Masing-masing baris berisi sebuah bilangan bulat sesuai penjelasan di atas.

Contoh Masukan

8

add 1 1

add 2 2

add 3 3

del 2

adx 2

del 2

dex 1

del 2

Contoh Keluaran

1

3

6

3

5

3

Penjelasan

Berikut ini adalah visualisasi dari contoh masukan.

awal: []

add 1 1: [1] -> cetak 1

add 2 2: [1 2 2] -> cetak 3

add 3 3: [1 2 2 3 3 3] -> cetak 6

del 2: [1 2 2 3] -> cetak 3

adx 2: [3 4 4 5]

del 2: [3 4] -> cetak 5

dex 1: [2 3]

del 2: [] -> cetak 3

Batasan

- $1 \leq Q \leq 1000$
- $1 \leq X, Y, D \leq 10\,000$

2. Umpan Balik dan Tindak Lanjut

1) Soal Mudah (*Stack*)

```
#include <iostream>
#include <stack>
#include <string>
using namespace std;

int main() {
    int N;
    cin >> N;
```

```

stack<string> piring;

// Operasi PUSH
cout << "Operasi PUSH" << endl;
for (int i = 1; i <= N; ++i) {
    string item = "Piring-" + to_string(i);
    piring.push(item);

    cout << "Item paling atas: " << piring.top()
<< " Jumlah item di Stack: " << piring.size() <<
endl;
}

// Operasi POP
cout << "\nOperasi POP" << endl;
while (!piring.empty()) {
    cout << "Item paling atas: " << piring.top()
<< "\n";

    cout << "Item yang di keluarkan: " <<
piring.top() << " Sisa item di Stack: " <<
piring.size() - 1;

    piring.pop();
    if (!piring.empty()) {
        cout << " Item paling atas: " <<
piring.top() << endl;
    } else {
        cout << endl;
    }
}

```

```
    }

    return 0;
}
```

2) Soal Mudah (Queue)

```
#include <iostream>
#include <queue>
#include <string>
using namespace std;

int main() {
    int N;
    cin >> N;

    queue<string> piring;
    // Operasi ENQUEUE
    cout << "Operasi ENQUEUE" << endl;
    for (int i = 1; i <= N; ++i) {
        string item = "Piring-" + to_string(i);
        piring.push(item);

        cout << "Item paling depan: " <<
piring.front()
            << " Jumlah item di Queue: " <<
piring.size() << endl;
    }

    // Operasi DEQUEUE
```

```
cout << "\nOperasi DEQUEUE" << endl;
while (!piring.empty()) {
    cout << "Item paling depan: " <<
piring.front() << endl;

    cout << "Item yang di keluarkan: " <<
piring.front() << " Sisa item di Queue: " <<
piring.size() - 1;

    piring.pop();

    if (!piring.empty()) {
        cout << " Item paling depan: " <<
piring.front() << endl;
    } else {
        cout << endl;
    }
}

return 0;
}
```



ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA



MODUL XI

Graph Berarah dan Tidak Berarah

**Elly Warni
Ingrid Nurtanio
Amil Ahmad Ilham
Adnan
Anugrayani Bustamin**

PRAKATA

Alhamdulillah, puji dan syukur kami panjatkan ke hadirat Allah SWT yang telah melimpahkan rahmat dan hidayah-Nya, sehingga modul ini dapat terselesaikan dengan baik. Shalawat serta salam senantiasa tercurah kepada junjungan kita, Nabi Muhammad SAW, beserta keluarga dan sahabat beliau yang telah membawa kita dari zaman kegelapan menuju terang benderang ilmu pengetahuan.

Modul ini disusun sebagai bagian dari rangkaian pembelajaran yang bertujuan untuk membantu mahasiswa memahami dan menerapkan konsep dasar struktur data graf berarah dan tidak berarah, serta dapat memanfaatkan graf untuk merepresentasikan dan menyelesaikan berbagai permasalahan terkait keterhubungan antar elemen. Modul ini merupakan lanjutan dari modul sebelumnya yang membahas dasar-dasar algoritma, sehingga diharapkan mahasiswa memiliki pemahaman yang lebih mendalam dan terstruktur.

Tidak dapat dipungkiri bahwa proses penyusunan modul ini tidak lepas dari dukungan berbagai pihak. Oleh karena itu, kami mengucapkan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan masukan dan bimbingan dalam proses penyusunan modul ini. Terima kasih kami sampaikan kepada rekan-rekan dosen dan staf yang telah memberikan kritik dan saran yang membangun, sehingga modul ini dapat terselesaikan dengan baik.

Kami menyadari bahwa modul ini masih memiliki keterbatasan dan kekurangan. Oleh karena itu, kami sangat mengharapkan masukan, kritik, dan saran dari para pembaca untuk penyempurnaan di masa yang akan datang. Semoga modul ini dapat memberikan manfaat dan menjadi bahan referensi yang berharga bagi para mahasiswa dalam memahami konsep algoritma dan pemrograman.

Akhir kata, semoga upaya ini mendapat ridha Allah SWT dan memberikan kontribusi positif bagi kemajuan ilmu pengetahuan. Amin.

Makassar, November 2024

Penulis

DAFTAR ISI

PRAKATA	i
DAFTAR ISI	iii
RPS MATA KULIAH ALGORITMA DAN STRUKTUR DATA	iv
MODUL ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA	10
Pengantar	10
KEGIATAN BELAJAR 1	12
A. Deskripsi Singkat	12
B. Relevansi	13
C. Pembelajaran	14
D. Penilaian Kegiatan Belajar	23

RPS MATA KULIAH ALGORITMA DAN STRUKTUR DATA



**UNIVERSITAS HASANUDDIN
FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK INFORMATIKA**

Kode
Dokumen

RENCANA PEMBELAJARAN SEMESTER

MATA KULIAH (MK)	KODE	Rumpun MK	BOBOT (sks)		SEMESTER	Tgl Penyusunan	
Algoritma Pemrograman dan Struktur Data	23D12110604	Teknik Informatika	T=2	P=2	2	19-02-2023	
OTORISASI	Pengembang RPS Dr. Ir. Ingrid Nurtanio., MT Dr. Amil Ahmad Ilham., ST., M.IT Adnan., ST., MT., PhD Anugrayani Bustamin., ST., MT Elly Warni, S.T., M.T	Koordinator RMK Dr. Ir. Ingrid Nurtanio., MT	Ketua PRODI Prof. Dr. Indrabayu., ST., M.T., M.Bus.Sys				
Capaian Pembelajaran (CP)	CPL-PRODI yang dibebankan pada MK <i>Intended Learning Outcomes</i>						
	CPL 1	Memiliki dasar pengetahuan Teknik Informatika yang meliputi teori dan konsep dasar dari Ilmu Komputer, Matematika dan Statistika, Algoritma dan Pemrograman, Rekayasa Perangkat Lunak, Manajemen Informasi dan Ketahanan Digital, serta pengetahuan tingkat lanjut pada bidang-bidang khusus Teknik Informatika, seperti Kecerdasan Buatan, Data Science, Jaringan Komputer, Komputasi Awan dan Internet of Things.					
	CPL 3	Mampu mengaplikasikan pengetahuan bidang Teknik Informatika yang dipadankan dengan bidang ilmu lainnya untuk menganalisa dan mencari solusi dari berbagai masalah berbasis komputasi.					
	CPL 4	Mampu mendesain, mengimplementasikan dan mengevaluasi solusi berbasis komputasi dengan mengaplikasikan ilmu Teknik Informatika dan dasar-dasar pembangunan perangkat lunak.					
	Capaian Pembelajaran Mata Kuliah (CPMK)						
	CPMK	Setelah mengikuti Mata Kuliah Algoritma Pemrograman dan Struktur Data selama 1 (satu) semester, mahasiswa mampu menguasai dan menerapkan konsep teoritis bidang Informatika dan ilmu komputer secara dan mendalam serta memformulasikan penyelesaian masalah procedural dengan pemikiran logis dan sistematis secara mandiri dan terukur sebagai dasar pengembangan perangkat lunak.					
CPL ⇒ Sub-CPMK							
	CPL 1	Sub-CPMK 1 : Mahasiswa mampu mendefinisikan konsep dan pengertian Algoritma					

	CPL 1	Sub-CPMK 2 : Mahasiswa mampu mendeksripsikan Dasar Algoritma
	CPL 3	Sub-CPMK 3 : Mahasiswa mampu mengimplementasikan Flowchart
	CPL 1	Sub-CPMK 4 : Mahasiswa mampu memahami bentuk pemilihan dan perulangan
	CPL 4	Sub-CPMK 5 : Mahasiswa mampu mengimplementasikan prosedur, fungsi dan pemrosesan teks
	CPL 1	Sub-CPMK 6 : Mahasiswa mampu mempraktekkan konsep array
	CPL 4	Sub-CPMK 7 : Mahasiswa mampu menetapkan ciri dari teknik pencarian
	CPL 4	Sub-CPMK 8 : Mahasiswa mampu menetapkan ciri dari teknik sorting dan menganalisis complexitas algoritma
	CPL 1	Sub-CPMK 9 : Mahasiswa mampu memahami struct dan konsep linked list
	CPL 1	Sub-CPMK 10 : Mahasiswa mampu memahami Antrian dan Stack
	CPL 1	Sub-CPMK 11 : Mahasiswa mampu memahami teknik graph
	CPL 3	Sub-CPMK 12 : Mahasiswa mampu mengimplementasi konsep Tree
Deskripsi Singkat MK	<p>Mata kuliah Algoritma Pemrograman dan Struktur Data merupakan mata kuliah yang menyajikan konsep pemrograman secara mendalam terkait algoritma dan struktur data. Muatan mata kuliah merupakan dasar dari ilmu matematika, algoritma dan logika yang menjadi syarat utama pada mata kuliah Pemrograman lanjut . Mata kuliah ini disajikan untuk mahasiswa semester dua di Program Studi Teknik Informatika, Penanggungjawab mata kuliah adalah Laboratorium Komputer . Metode Pembelajaran bauran, teori dan praktik</p>	
Bahan Kajian / Materi Pembelajaran	<ol style="list-style-type: none"> 1. Konsep Algoritma 2. Dasar Algoritma (Aturan Penulisan Header, deklarasi dan deskripsi) 3. Flowchart (Simbol, variabel, algoritma cabang, runtunan) 4. Pemilihan (Alir Kontrol dan Pengulangan) 5. Prosedur, Fungsi dan Pemrosesan Teks 6. Array – Larik 7. Pencarian (Binary Search) 8. Pencarian (Sequential Search) 9. Bubble dan Insertion Sort 10. Merge dan selection sort 11. Quick sort dan complexita algoritma 12. Struct dan Linked List 13. Antrian dan Stack 14. Graph berarah dan tidak berarah 15. Tree 	

Pustaka	Utama :	<ol style="list-style-type: none"> 1. Munir, Rinaldi dan Lidya, Leony. 2016. Algoritma dan Pemrograman dalam Bahasa Pascal, C dan C++ Edisi Keenam. Informatika : Bandung 2. Gozali, William dan Aji, Alham Fikri. 2014. Pemrograman Kompetitif Dasar. Ikatan Alumni Tim Olimpiade Komputer Indonesia. 3. Zakaria, Teddy M dan Prijono, Agus. 2005. Konsep dan Implementasi Struktur Data. Informatika : Bandung. 4. Niswar, M., Ilham, A., Zainuddin, Z., Adnan, A., P, A., Warni, E., Aswad, I., & Muslimin, Z. (2021). Sosialisasi Metode Berfikir Komputasional pada Pendidikan Dasar dan Menengah di Lingkup Sulawesi Selatan. URNAL EPAT eknologi erapan ntuk engabdian asyarakat, 4(1), 46-52. https://doi.org/10.25042/jurnal_tepat.v4i1.172 						
	Pendukung:							
	Tuliskan pustaka pendukung jika ada, sebagai pengayaan literasi							
Dosen Pengampu		<ol style="list-style-type: none"> 1. Dr. Ir. Ingrid Nurtanio., MT 2. Dr. Amil Ahmad Ilham., ST., M.IT 3. Adnan., ST., MT., PhD 4. Anugrayani Bustamin., ST., MT 5. Elly Warni, S.T., M.T 						
Matakuliah syarat		Dasar Pemrograman Komputer						
				Bentuk Pembelajaran,				
Pekan Ke-	Sub-CPMK (Kemampuan akhir tiap tahapan belajar)	Penilaian		Metode Pembelajaran, Penugasan Mahasiswa, [Estimasi Waktu]	Materi Pembelajaran [Pustaka]		Bobot Penilaian (%)	
		Indikator	Kriteria & Bentuk	Luring (offline)			Daring (online)	
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	

1	Mahasiswa mampu mengetahui konsep dan pengertian Algoritma	Mengetahui konsep definisi, manfaat, mekanisme dan kontribusi Algoritma	Assignments	Lecture TM: 4 x 50 menit.	Lecture VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Konsep dan definisi algoritma 2. Mekanisme dan Kontribusi Algoritma	5%
2	Mahasiswa mampu memahami Dasar Algoritma CPL 3- SubCPMK 2	Memahami Dasar Algoritma (tipe data, aturan penulisan, proses, instruksi dan aksi)	quizzes	Lecture, case study TM: 4 x 50 menit.	Lecture, case study VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Dasar Algoritma (tipe data, aturan penulisan, proses, instruksi, dan aksi) Pustaka : [1]	5%
3	Mahasiswa mampu mengenali Flowchart CPL 4- SubCPMK 3	Mampu mengenali Flowchart (simbol, variabel, algoritma cabang, dan runtunan)	Assignments	Lecture, small group discussion TM: 4 x 50 menit.	Lecture, small group discussion VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit	Flowchart (Simbol, variabel, algoritma cabang, runtunan) Pustaka : [1]	7%

4	Mahasiswa mampu memahami bentuk pemilihan (alih kontrol dan pengulangan)	Mampu memahami bentuk pemilihan (alih kontrol dan pengulangan)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Pemilihan Kontrol Pengulangan) Pustaka : [1]	(Alir dan 7%
5	Mahasiswa mampu mengimplementasikan prosedur, fungsi dan pemrosesan teks	Mampu mengimplementasikan pemanggilan prosedur, pemanggilan fungsi dan pemrosesan teks	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Pemanggilan Prosedur 2. Pemanggilan Fungsi 3. Pemrosesan Teks Pustaka : [1]	7%
6	Mahasiswa mampu mempraktekkan konsep array	Mampu mempraktekkan konsep array seperti menginisialisasi array, mengisi elemen, mencari	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Menginisialisasi array 2. Mengisi elemen array 3. Mencari nilai maksimum dan minimum	7%

		nilai maksimum dan minimum			/104D4224/inde x.php?id_session =13610	Pustaka : [1][2]	
7	Mahasiswa mampu menetapkan ciri dari teknik pencarian	Mampu menetapkan ciri dari teknik pencarian ● Binary Search ● Sequential Search	Assignments quizzes	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Binary Search 2. Sequential Search Pustaka : [2][3]	7%
8	Evaluasi Tengah Semester / Ujian Tengah Semester						
9	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting ● Bubble Sort ● Insertion Sort	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Bubble Sort 2. Insertion Sort Pustaka : [2][3]	9%
	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting Merge Sort	Assignments	Lecture, practicum TM: 4 x 50	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit.	1. Merge Sort Selection Sort Pustaka : [2][3]	9%

10		Selection Sort		menit.	BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610		
11	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting Quick Sort berserta complexitas Algoritmanya	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Quick Sort 2. Complexitas Algoritma Pustaka : [2][3]	9%
12	Mahasiswa mampu memahami struct dan konsep linked list	Mampu memahami struct dan konsep dan operasi dasar linked list (deklarasi simpul, penambahan simpul pada linked list awal, akhir dan tengah)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Konsep Struct dan implementasinya 2. Deklarasi list 3. Penambahan simpul pada list awal, akhir dan tengah Pustaka : [2][3]	7%

13	Mahasiswa mampu memahami Antrian dan Stack	Mampu memahami Antrian dan Stack dalam List	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Antrian dan Stack dalam list Pustaka : [2][3]	7%
14	Mahasiswa mampu memahami teknik graph	mampu memahami teknik graph (graph berarah dan tidak berarah menggunakan linked list)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Graph berarah Graph tidak berarah Pustaka : [2][3]	7%
15	Mahasiswa mampu mengimplementasi konsep Tree	Mampu mengimplementasi konsep Tree menggunakan linked list	Assignments quizzes	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit	Konsep tree menggunakan linked List Pustaka : [2][3]	7%

					https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610		
16	Evaluasi Akhir Semester / Ujian Akhir Semester						

MODUL ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA

MODUL XI

GRAPH BERARAH DAN TIDAK BERARAH

Pengantar

Modul ini dirancang untuk membantu mahasiswa memahami dasar-dasar graf berarah dan tidak berarah, termasuk konsep simpul, sisi, serta representasi graf yang digunakan dalam pemrograman. Dalam rangka mencapai capaian pembelajaran mata kuliah (CPMK) yang telah ditetapkan, mahasiswa akan diarahkan untuk memahami dan mengidentifikasi komponen penting dalam pemodelan graf serta penerapannya.

Pada Sub-CPMK kali ini, mahasiswa diharapkan mampu mengenali dan memahami perbedaan antara graf berarah dan tidak berarah serta bagaimana kedua jenis graf tersebut digunakan untuk memodelkan berbagai jenis masalah dalam pemrograman. Kegiatan belajar akan dilakukan melalui diskusi kelompok kecil dan pembelajaran melalui ceramah serta didukung oleh platform pembelajaran daring.

Kegiatan pembelajaran ini akan mencakup,

- Tatap Muka (TM): 4 sesi, masing-masing 50 menit.
- Virtual Classroom (VC): 4 sesi, masing-masing 50 menit.
- Praktikum Terbimbing (PT): 4 sesi, masing-masing 60 menit.
- Bimbingan Mandiri (BM): 4 sesi, masing-masing 60 menit

Sumber pembelajaran yang dapat diakses untuk mendukung pemahaman lebih lanjut tersedia di platform sikola, dengan pustaka referensi yang telah disediakan.

Setelah mengikuti modul ini, mahasiswa diharapkan dapat mencapai tingkat pemahaman yang cukup, setidaknya 80%, sebelum melanjutkan ke materi berikutnya

KEGIATAN BELAJAR 1

GRAPH BERARAH DAN TIDAK BERARAH

A. Deskripsi Singkat

Mahasiswa yang mempelajari graf berarah dan tidak berarah sering menghadapi tantangan dalam memahami konsep dasar dan penggunaan struktur graf ini dalam berbagai konteks. Pemahaman graf penting karena graf adalah struktur data fundamental yang digunakan untuk merepresentasikan hubungan antar objek, baik dalam jaringan komputer, sistem transportasi, maupun aplikasi sosial.

Dalam kegiatan belajar pertama ini, mahasiswa akan diperkenalkan pada konsep dasar graf berarah dan tidak berarah serta jenis-jenis yang ada pada graf. Graf berarah menunjukkan koneksi satu arah antara simpul-simpulnya, sedangkan graf tidak berarah menggambarkan hubungan timbal balik. Mahasiswa akan belajar tentang komponen utama graf seperti simpul (node) dan sisi (edge).

Teori yang akan disampaikan mencakup berbagai aplikasi praktis dari graf, misalnya dalam algoritma pencarian rute dan pemodelan jaringan. Mahasiswa akan diajarkan cara menggambarkan struktur graf dan mempelajari teknik dasar dalam pemrosesan data berbasis graf.

Dengan pemahaman yang baik, mahasiswa diharapkan dapat memanfaatkan graf berarah dan tidak berarah untuk menyelesaikan masalah dalam pemrograman dan algoritma yang memerlukan pemetaan hubungan antar komponen atau objek.

B. Relevansi

Materi dalam modul sebelumnya membahas struktur data *stack* dan *queue*, yang merupakan dasar dalam pengelolaan data secara teratur, terutama untuk implementasi antrian dan proses bertumpuk. Pemahaman tentang *stack* dan *queue* sangat bermanfaat saat mahasiswa mulai mempelajari algoritma graf berarah dan tidak berarah, karena algoritma-algoritma yang sering diterapkan pada graf, seperti *Depth-First Search* (DFS) dan *Breadth-First Search* (BFS), memanfaatkan struktur *stack* dan *queue* dalam proses pencarian atau penjelajahan simpul-simpul graf.

Pemahaman graf berarah dan tidak berarah sangat penting karena graf digunakan untuk memodelkan banyak masalah nyata, seperti jaringan transportasi dan rute navigasi GPS. Dalam modul ini, mahasiswa diperkenalkan pada konsep dasar graf, termasuk simpul, sisi, arah, dan bagaimana struktur graf dapat memetakan hubungan kompleks antar elemen. Dengan landasan ini, mahasiswa dapat memahami dan menerapkan graf untuk menganalisis relasi antar data secara lebih terstruktur dan visual.

Secara keseluruhan, pemahaman graf dalam rangkaian modul ini sangat relevan terhadap pencapaian Sub-Capaian Pembelajaran Mata Kuliah (Sub-CPMK). Modul ini melatih kemampuan mahasiswa dalam menggunakan graf untuk menyelesaikan masalah dalam pemrograman dan mempersiapkan mereka untuk mempelajari algoritma lainnya. Mahasiswa diharapkan dapat mengasah kemampuan dalam menganalisis masalah serta berpikir logis untuk merancang solusi pemrograman yang efisien dan terstruktur dengan pendekatan berbasis graf.

C. Pembelajaran

Capaian pembelajaran yang diharapkan pada kegiatan belajar yang pertama ini adalah peserta kuliah mampu memahami dan menerapkan konsep dasar struktur data graf berarah dan tidak berarah, serta dapat memanfaatkan graf untuk merepresentasikan dan menyelesaikan berbagai permasalahan terkait keterhubungan antar elemen. Untuk mencapai kompetensi lulusan ini, peserta kuliah diharapkan membaca dan menyelesaikan Kegiatan Belajar 1 ini. Untuk mencapai capaian pembelajaran pada Kegiatan Belajar 1, peserta kuliah harus mengikuti urutan kegiatan pembelajaran yang telah disiapkan dalam modul ini.

1. Uraian

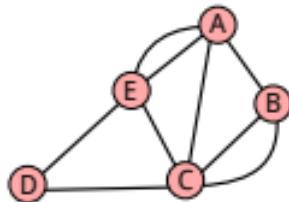
1.1 Konsep Graf

Graf merupakan konsep matematika diskrit untuk merepresentasikan objek-objek yang saling berhubungan. Secara informal, suatu graf adalah himpunan benda-benda yang disebut "simpul" (vertex atau node) yang terhubung oleh "sisi" (edge) atau "busur" (arc). Biasanya graf digambarkan sebagai kumpulan titik-titik (melambangkan "simpul") yang dihubungkan oleh garis-garis atau garis berpanah. Objek ini bisa saja berupa kota-kota yang terhubung dengan ruas jalan, individu-individu manusia yang saling mengenal, atau hubungan mangsa-pemangsa antar hewan di alam. Melalui pemahaman tentang konsep graf, kita dapat mendekati permasalahan yang melibatkan hubungan antar objek dengan lebih baik.

Graf adalah struktur yang terdiri dari:

- Node /vertex, yaitu objek yang merepresentasikan suatu konsep.
- Edge, yaitu penghubung antar dua node.

Sebagai contoh, misalkan kita ingin memodelkan sejumlah kota yang dihubungkan oleh beberapa ruas jalan dengan graf. Kita dapat menganggap setiap kota sebagai node, dan ruas jalan sebagai edge.



Terdapat lima kota, yaitu A, B, C, D, dan E.

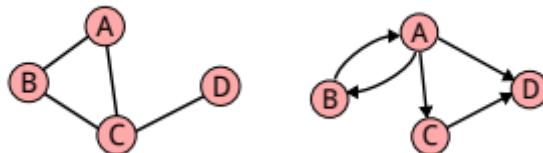
Terdapat sembilan ruas jalan, yaitu:

1. Antara kota A dengan kota B.
2. Antara kota A dengan kota C.
3. Antara kota A dengan kota E, sebanyak dua ruas.
4. Antara kota B dengan kota C, sebanyak dua ruas.
5. Antara kota C dengan kota D.
6. Antara kota C dengan kota E.
7. Antara kota D dengan kota E.

1.2 Jenis Graf

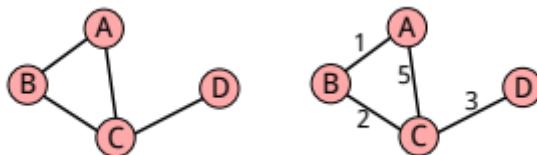
Berdasarkan hubungan antar node, graf dapat dibedakan menjadi:

- Graf tak berarah: edge dari A ke B dapat ditelusuri dari A ke B dan B ke A.
- Graf berarah: edge dari A ke B hanya dapat ditelusuri dari A ke B.

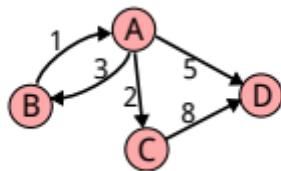


Sementara berdasarkan bobot dari edge, graf dapat dibedakan menjadi:

- Graf tak berbobot, yaitu graf dengan edge yang bobotnya seragam dan hanya bermakna terdapat hubungan antar node.
- Graf berbobot, yaitu graf dengan edge yang dapat memiliki bobot berbeda-beda. Bobot pada edge ini bisa jadi berupa biaya, jarak, atau waktu yang harus ditempuh jika menggunakan edge tersebut.



Tentu saja, suatu graf dapat memiliki kombinasi dari sifat-sifat tersebut. Misalkan graf yang berbobot dan berarah.



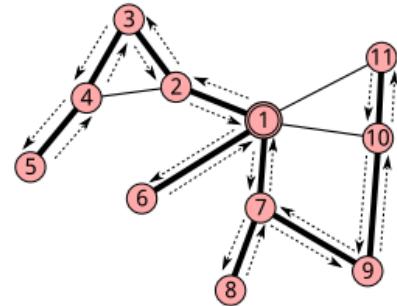
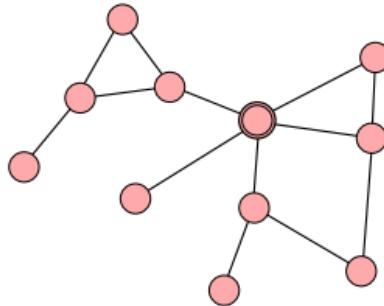
1.3 Penjelajahan Graf

Penjelajahan graf merupakan penelusuran node-node pada suatu graf menurut suatu aturan tertentu. Terdapat 2 metode yang dapat digunakan, yaitu DFS dan BFS.

DFS: Depth-First Search

Penelusuran dilakukan dengan mengutamakan node yang lebih dalam terlebih dahulu (depth-first). Misalkan penelusuran secara DFS dimulai dari suatu node. Selama masih terdapat tetangga yang belum dikunjungi, kunjungi tetangga tersebut. Pada node tetangga ini, lakukan pula hal yang serupa. Ketika kita mencapai

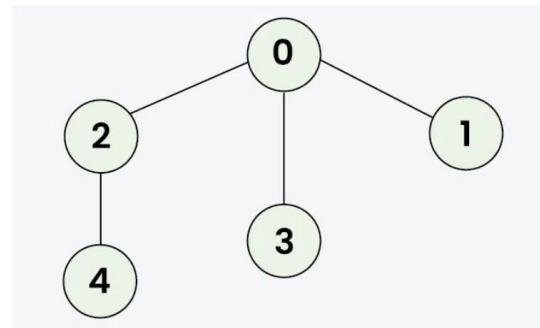
suatu node yang mana seluruh tetangganya sudah pernah dikunjungi, kembali ke node terakhir yang dikunjungi tepat sebelum node ini dikunjungi



Dalam pemrograman, DFS biasa dilakukan dengan rekursi atau struktur data stack. Untuk keperluan implementasi, misalkan:

- Terdapat V node dan E edge.
- Setiap node dinomori dari 1 sampai dengan V .
- $\text{adj}(x)$ menyatakan himpunan tetangga dari node x .
- Terdapat array visited, yang mana $\text{visited}[x]$ bernilai true hanya jika x telah dikunjungi.

Sebagai contoh bisa dilihat gambar dan kode dibawah ini:



$\text{adj} = [[2,3,1], [0], [0,4], [0], [2]]$

Input:

start = 0

Output:

0 2 4 3 1

Penjelasan:

- a. **Mulai dari simpul 0:** Tandai sebagai dikunjungi. Output: 0
- b. **Pindah ke simpul 2:** Tandai sebagai dikunjungi. Output: 2
- c. **Pindah ke simpul 4:** Tandai sebagai dikunjungi. Output: 4
(kemudian kembali ke simpul 2, lalu kembali ke simpul 0)
- d. **Pindah ke simpul 3:** Tandai sebagai dikunjungi. Output: 3
(kemudian kembali ke simpul 0)
- e. **Pindah ke simpul 1:** Tandai sebagai dikunjungi. Output: 1

Code (Penelusuran graf dengan DFS secara rekursif):

```
#include <iostream>
#include <vector>
using namespace std;

vector<vector<int>> adj = {{2, 3, 1}, {0}, {0, 4},
{0}, {2}}; // daftar tetangga dari setiap simpul
vector<bool> visited(5, false); // vektor untuk
menandai simpul yang sudah dikunjungi

void DFS(int v) {
    visited[v] = true; // tandai simpul v sebagai
dikunjungi
    cout << v << " "; // cetak simpul yang
dikunjungi

    // untuk setiap tetangga u dari v yang belum
dikunjungi, lakukan DFS
    for (int u : adj[v]) {
        if (!visited[u]) {
            DFS(u);
        }
    }
}

int main() {
    DFS(0); // mulai dari simpul 0
}
```

Code (Penelusuran graf dengan DFS secara iteratif menggunakan stack):

```
#include <iostream>
#include <vector>
#include <stack>
using namespace std;

vector<vector<int>> adj = {{1, 3, 2}, {0}, {0, 4}, {0}, {2}}; // daftar tetangga dari setiap simpul
vector<bool> visited(5, false); // vektor untuk menandai simpul yang sudah dikunjungi
stack<int> s;

void DFS(int start) {
    s.push(start);
    visited[start] = true;

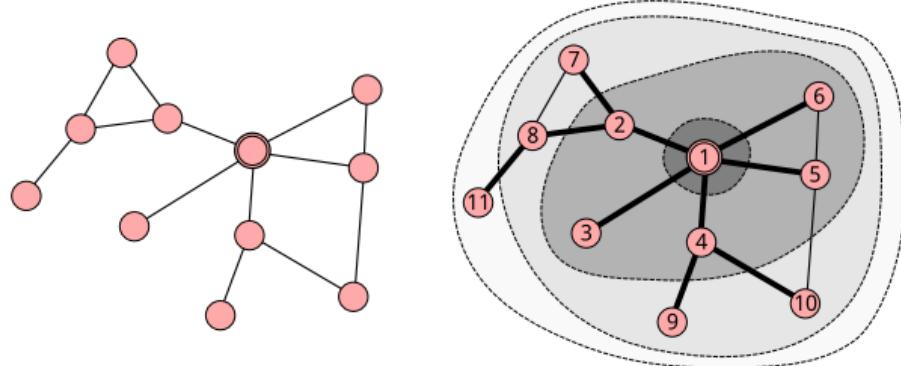
    while (!s.empty()) {
        int v = s.top();
        s.pop();
        cout << v << " "; // cetak simpul yang sedang dikunjungi

        // periksa tetangga-tetangga dari simpul v
        for (int u : adj[v]) {
            if (!visited[u]) {
                s.push(u);
                visited[u] = true; // tandai sebagai dikunjungi
            }
        }
    }
}

int main() {
    int start = 0; // mulai dari simpul 0
    DFS(start);    // panggil fungsi DFS dari simpul awal
}
```

BFS: Breadth-First Search

Pada BFS, penelusuran node pada graf dilakukan lapis demi lapis. Semakin dekat suatu node dengan node awal, node tersebut akan dikunjungi terlebih dahulu.



Dalam pemrograman, BFS biasa diimplementasikan dengan bantuan struktur data queue. Queue ini menyimpan daftar node yang akan dikunjungi. Untuk setiap fase, kunjungi node yang terdapat di paling depan queue, dan daftarkan seluruh tetangganya yang belum dikunjungi pada bagian akhir queue. Lakukan hal ini sampai seluruh node terkunjungi.

Contoh kode C++:

```
// Program C++ untuk BFS dari graf tidak berarah
#include <iostream>
#include <queue>
#include <vector>
using namespace std;

// BFS dari sumber yang diberikan s
void bfs(vector<vector<int>>& adj, int s)
{
    // Buat antrian untuk BFS
    queue<int> q;

    // Awalnya tandai semua simpul sebagai belum
    // dikunjungi
    // Ketika kita memasukkan simpul ke dalam
    // antrian, kita tandai sebagai
```

```

// telah dikunjungi
vector<bool> visited(adj.size(), false);

// Tandai simpul sumber sebagai dikunjungi dan
// masukkan ke dalam antrian
visited[s] = true;
q.push(s);

// Iterasi di atas antrian
while (!q.empty()) {

    // Ambil simpul dari antrian dan cetak
    int curr = q.front();
    q.pop();
    cout << curr << " ";

    // Dapatkan semua simpul yang berdekatan
    dengan simpul yang diambil
    // Jika simpul yang berdekatan belum
    dikunjungi, tandai sebagai dikunjungi dan masukkan
    ke dalam antrian
    for (int x : adj[curr]) {
        if (!visited[x]) {
            visited[x] = true;
            q.push(x);
        }
    }
}

// Fungsi untuk menambahkan edge ke graf
void addEdge(vector<vector<int>>& adj, int u, int v)
{
    adj[u].push_back(v);
    adj[v].push_back(u); // Graf tidak berarah
}

int main()
{

```

```

// Jumlah simpul dalam graf
int V = 5;

// Representasi daftar adjacency dari graf
vector<vector<int>> adj(V);

// Tambahkan edge ke graf
addEdge(adj, 0, 1);
addEdge(adj, 0, 2);
addEdge(adj, 1, 3);
addEdge(adj, 1, 4);
addEdge(adj, 2, 4);

// Lakukan traversal BFS mulai dari simpul 0
cout << "BFS mulai dari 0 : \n";
bfs(adj, 0);

return 0;
}

```

Analisis Kompleksitas DFS dan BFS

Baik DFS maupun BFS sama-sama mengunjungi setiap node tepat satu kali, dengan memanfaatkan seluruh edge.

Kompleksitas dari kedua metode adalah:

- $O(V^2)$, jika digunakan adjacency matrix.
- $O(V + E)$, jika digunakan adjacency list.

2. Rangkuman

Graf adalah struktur matematika yang merepresentasikan objek-objek yang saling terhubung, seperti kota-kota yang dihubungkan dengan ruas jalan. Graf terdiri dari node (vertex) yang mewakili objek dan edge yang menjadi penghubung antar node. Berdasarkan sifat edge, graf dapat dibedakan menjadi graf tak berarah, di mana edge dapat ditempuh dalam dua arah, dan graf berarah, di mana edge hanya dapat ditempuh satu arah. Selain itu, graf juga dapat berbobot,

di mana setiap edge memiliki bobot yang menunjukkan jarak atau biaya, atau tak berbobot, di mana edge hanya menunjukkan adanya hubungan. Penjelajahan graf dapat dilakukan dengan dua metode, yaitu Depth-First Search (DFS) yang menelusuri node hingga kedalaman maksimal sebelum kembali, serta Breadth-First Search (BFS) yang menjelajahi node secara lapis demi lapis. DFS dapat diimplementasikan dengan rekursi atau stack, sementara BFS menggunakan queue untuk melacak node yang akan dikunjungi. Kedua algoritma ini memiliki kompleksitas waktu $O(V + E)$ jika menggunakan adjacency list, menjadikannya penting dalam menyelesaikan berbagai masalah yang melibatkan hubungan antar objek.

3. Pustaka

- a. Aji, Alham Fikri dan Gozali, William. *Pemrograman Kompetitif Dasar*. ISBN 978-602-6598-89-9.
- b. Muliono, R. (2019). *Pemrograman C++: Algoritma & Struktur Data*.

D. Penilaian Kegiatan Belajar

1. Penilaian

1.1 Latihan

1) Perjalanan Tersingkat

Deskripsi

Di suatu provinsi, terdapat N buah kota. Kota-kota dinomori dari 1 sampai dengan N. Terdapat E ruas jalan yang masing-masing menghubungkan dua kota berbeda. Pak Dengklek tinggal di kota A. Suatu hari, beliau ingin pergi ke kota B. Karena sudah tua, Pak Dengklek ingin melewati sesedikit

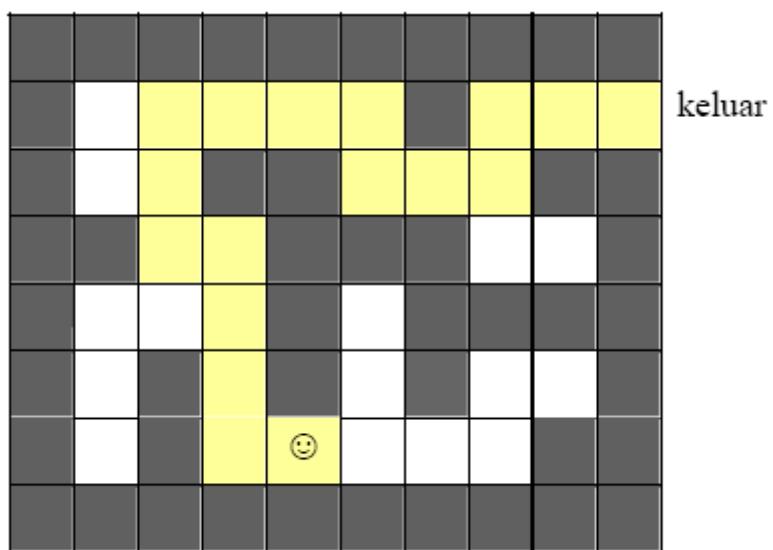
mungkin ruas jalan untuk sampai ke kota B. Tentukan berapa banyak ruas jalan tersedikit yang perlu beliau lewati untuk pergi dari kota A ke kota B! (Dijamin bahwa kota A akan selalu bisa sampai di kota B)

1.2 Tantangan

- 2) Maze (Sumber <https://tlx.toki.id/problems/osn-2005/2B>)

Deskripsi

Anda tahu permainan maze? Kalau melihat gambar berikut ini pasti tahu. Dalam permainan maze ini makhluk yang digambarkan dengan bulatan wajah Mr. Groovy harus mencari jalan ke luar dari petak maze yang diberikan.



Jalan keluar yang harus dilaluinya adalah kota-kotak yang berwarna kuning tersebut. Masalahnya karena bisa terdapat beberapa cara untuk mencapai jalan keluar maka di sini Anda harus menemukan jumlah kotak yang paling sedikit dalam lintasan, yang menyatakan juga jumlah langkah terpendek untuk mencapai bagian luar. Dalam hal contoh maze di atas yang paling sedikit adalah 17 kotak/langkah yaitu yang berwana kuning tersebut.

Diberikan maze yang berukuran $N \times M$, dengan posisi awal Mr. Groovy pada (A, B) , tentukan jumlah langkah minimum yang dibutuhkan untuk mencapai ke luar.

Format Masukan

Baris pertama berisi dua buah bilangan bulat N dan M . N baris berikutnya masing-masing berisi M buah bilangan -1 yang menyatakan dinding yang tidak dapat ditembus, atau 0 yang menyatakan ruang yang dapat dilalui. Baris berikutnya berisi dua buah bilangan bulat A dan B . Dijamin petak (A, B) adalah ruangan. Dijamin sekurangnya ada satu jalan keluar.

Format Keluaran

Sebuah baris berisi sebuah bilangan bulat yaitu langkah minimum untuk mencapai ke luar.

Contoh Masukan

```
8 10
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 0 0 0 0 0 -1 0 0 0
-1 0 0 -1 -1 0 0 0 -1 -1
-1 -1 0 0 -1 -1 -1 0 0 -1
-1 0 0 0 -1 0 -1 -1 -1 -1
-1 0 -1 0 -1 0 -1 0 0 -1
-1 0 -1 0 0 0 0 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1
```

7 5

Contoh Keluaran

17

Batasan

- $3 \leq N, M \leq 100$
- $1 \leq A \leq N$

$$1 \leq B \leq M$$

2. Umpam Balik dan Tindak Lanjut

1) Source Code Latihan (Perjalanan Tersingkat)

```
#include <bits/stdc++.h>
using namespace std;

// Untuk set jarak tiap kota
const int INF = INT_MAX-1;

int main() {
    int N, E, A, B;

    cin >> N >> E;
    vector<pair<int, int>> edges(E);

    vector<vector<int>> adj(N + 1);

    // Buat adjacency list untuk graf
    for (int i = 0; i < E; i++) {
        int u, v;
        cin >> u >> v;
        // edges[i] = {u, v};
        adj[u].push_back(v);
        adj[v].push_back(u); // Karena graf tak berarah
    }

    cin >> A >> B;

    // Inisialisasi vektor jarak dengan nilai tak terhingga (N + 1 karena penomoran dari 1 hingga N)
    vector<int> distance(N + 1, INF);
    queue<int> q;

    // Mulai dari kota A
    distance[A] = 0;
```

```

q.push(A);

// Lakukan BFS
while (!q.empty()) {
    int city = q.front();
    q.pop();

    // Periksa semua tetangga dari city
    for (int neighbor : adj[city]) {
        if (distance[neighbor] > distance[city]
+ 1) { // Jika jarak kota A + 1 lebih kecil dari
jarak kota B
            distance[neighbor] = distance[city]
+ 1; // Tambah jarak 1 dari kota sekarang
            q.push(neighbor);
        }
    }
}

// Keluarkan nilai jarak kota B terkecil
cout << distance[B];
return 0;
}

```



ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA



MODUL XII

TREE

*Elly Warni
Ingrid Nurtanio
Amil Ahmad Ilham
Adnan
Anugrayani Bustamin*

PRAKATA

Alhamdulillah, puji dan syukur kami panjatkan ke hadirat Allah SWT yang telah melimpahkan rahmat dan hidayah-Nya, sehingga modul ini dapat terselesaikan dengan baik. Shalawat serta salam senantiasa tercurah kepada junjungan kita, Nabi Muhammad SAW, beserta keluarga dan sahabat beliau yang telah membawa kita dari zaman kegelapan menuju terang benderang ilmu pengetahuan.

Modul ini disusun sebagai bagian dari rangkaian pembelajaran yang bertujuan untuk membantu mahasiswa memahami konsep dasar tree sebagai struktur data yang terhubung tanpa cycle, serta dapat menjelaskan hubungan antara jumlah node dan edge dalam tree. Modul ini merupakan lanjutan dari modul sebelumnya yang membahas dasar-dasar algoritma, sehingga diharapkan mahasiswa memiliki pemahaman yang lebih mendalam dan terstruktur.

Tidak dapat dipungkiri bahwa proses penyusunan modul ini tidak lepas dari dukungan berbagai pihak. Oleh karena itu, kami mengucapkan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan masukan dan bimbingan dalam proses penyusunan modul ini. Terima kasih kami sampaikan kepada rekan-rekan dosen dan staf yang telah memberikan kritik dan saran yang membangun, sehingga modul ini dapat terselesaikan dengan baik.

Kami menyadari bahwa modul ini masih memiliki keterbatasan dan kekurangan. Oleh karena itu, kami sangat mengharapkan masukan, kritik, dan saran dari para pembaca untuk penyempurnaan di masa yang akan datang. Semoga modul ini dapat memberikan manfaat dan menjadi bahan referensi yang berharga bagi para mahasiswa dalam memahami konsep algoritma dan pemrograman.

Akhir kata, semoga upaya ini mendapat ridha Allah SWT dan memberikan kontribusi positif bagi kemajuan ilmu pengetahuan. Amin.

Makassar, November 2024

Penulis

DAFTAR ISI

PRAKATA	i
DAFTAR ISI	iii
RPS MATA KULIAH ALGORITMA DAN STRUKTUR DATA	iv
MODUL ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA	10
Pengantar	10
KEGIATAN BELAJAR 1	12
A. Deskripsi Singkat	12
B. Relevansi	13
C. Pembelajaran	14
D. Penilaian Kegiatan Belajar	18

RPS MATA KULIAH ALGORITMA DAN STRUKTUR DATA



**UNIVERSITAS HASANUDDIN
FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK INFORMATIKA**

Kode Dokumen

RENCANA PEMBELAJARAN SEMESTER

MATA KULIAH (MK)	KODE	Rumpun MK	BOBOT (sks)		SEMESTER	Tgl Penyusunan			
Algoritma Pemrograman dan Struktur Data	23D12110604	Teknik Informatika	T=2	P=2	2	19-02-2023			
OTORISASI	Pengembang RPS Dr. Ir. Ingrid Nurtanio., MT Dr. Amil Ahmad Ilham., ST., M.IT Adnan., ST., MT., PhD Anugrayani Bustamin., ST., MT Elly Warni, S.T., M.T	Koordinator RMK Dr. Ir. Ingrid Nurtanio., MT	Ketua PRODI Prof. Dr. Indrabayu., ST., M.T., M.Bus.Sys						
Capaian Pembelajaran (CP)	CPL-PRODI yang dibebankan pada MK <i>Intended Learning Outcomes</i>								
	CPL 1	Memiliki dasar pengetahuan Teknik Informatika yang meliputi teori dan konsep dasar dari Ilmu Komputer, Matematika dan Statistika, Algoritma dan Pemrograman, Rekayasa Perangkat Lunak, Manajemen Informasi dan Ketahanan Digital, serta pengetahuan tingkat lanjut pada bidang-bidang khusus Teknik Informatika, seperti Kecerdasan Buatan, Data Science, Jaringan Komputer, Komputasi Awan dan Internet of Things.							
	CPL 3	Mampu mengaplikasikan pengetahuan bidang Teknik Informatika yang dipadankan dengan bidang ilmu lainnya untuk menganalisa dan mencari solusi dari berbagai masalah berbasis komputasi.							
	CPL 4	Mampu mendesain, mengimplementasikan dan mengevaluasi solusi berbasis komputasi dengan mengaplikasikan ilmu Teknik Informatika dan dasar-dasar pembangunan perangkat lunak.							
	Capaian Pembelajaran Mata Kuliah (CPMK)								
	CPMK	Setelah mengikuti Mata Kuliah Algoritma Pemrograman dan Struktur Data selama 1 (satu) semester, mahasiswa mampu menguasai dan menerapkan konsep teoritis bidang Informatika dan ilmu komputer secara dan mendalam serta memformulasikan penyelesaian masalah procedural dengan pemikiran logis dan sistematis secara mandiri dan terukur sebagai dasar pengembangan perangkat lunak.							
CPL ⇒ Sub-CPMK									
	CPL 1	Sub-CPMK 1 : Mahasiswa mampu mendefinisikan konsep dan pengertian Algoritma							

	CPL 1	Sub-CPMK 2 : Mahasiswa mampu mendeksripsikan Dasar Algoritma
	CPL 3	Sub-CPMK 3 : Mahasiswa mampu mengimplementasikan Flowchart
	CPL 1	Sub-CPMK 4 : Mahasiswa mampu memahami bentuk pemilihan dan perulangan
	CPL 4	Sub-CPMK 5 : Mahasiswa mampu mengimplementasikan prosedur, fungsi dan pemrosesan teks
	CPL 1	Sub-CPMK 6 : Mahasiswa mampu mempraktekkan konsep array
	CPL 4	Sub-CPMK 7 : Mahasiswa mampu menetapkan ciri dari teknik pencarian
	CPL 4	Sub-CPMK 8 : Mahasiswa mampu menetapkan ciri dari teknik sorting dan menganalisis complexitas algoritma
	CPL 1	Sub-CPMK 9 : Mahasiswa mampu memahami struct dan konsep linked list
	CPL 1	Sub-CPMK 10 : Mahasiswa mampu memahami Antrian dan Stack
	CPL 1	Sub-CPMK 11 : Mahasiswa mampu memahami teknik graph
	CPL 3	Sub-CPMK 12 : Mahasiswa mampu mengimplementasi konsep Tree
Deskripsi Singkat MK	<p>Mata kuliah Algoritma Pemrograman dan Struktur Data merupakan mata kuliah yang menyajikan konsep pemrograman secara mendalam terkait algoritma dan struktur data. Muatan mata kuliah merupakan dasar dari ilmu matematika, algoritma dan logika yang menjadi syarat utama pada mata kuliah Pemrograman lanjut . Mata kuliah ini disajikan untuk mahasiswa semester dua di Program Studi Teknik Informatika, Penanggungjawab mata kuliah adalah Laboratorium Komputer . Metode Pembelajaran bauran, teori dan praktik</p>	
Bahan Kajian / Materi Pembelajaran	<ol style="list-style-type: none"> 1. Konsep Algoritma 2. Dasar Algoritma (Aturan Penulisan Header, deklarasi dan deskripsi) 3. Flowchart (Simbol, variabel, algoritma cabang, runtunan) 4. Pemilihan (Alir Kontrol dan Pengulangan) 5. Prosedur, Fungsi dan Pemrosesan Teks 6. Array – Larik 7. Pencarian (Binary Search) 8. Pencarian (Sequential Search) 9. Bubble dan Insertion Sort 10. Merge dan selection sort 11. Quick sort dan complexita algoritma 12. Struct dan Linked List 13. Antrian dan Stack 14. Graph berarah dan tidak berarah 15. Tree 	

Pustaka	Utama :	<ol style="list-style-type: none"> 1. Munir, Rinaldi dan Lidya, Leony. 2016. Algoritma dan Pemrograman dalam Bahasa Pascal, C dan C++ Edisi Keenam. Informatika : Bandung 2. Gozali, William dan Aji, Alham Fikri. 2014. Pemrograman Kompetitif Dasar. Ikatan Alumni Tim Olimpiade Komputer Indonesia. 3. Zakaria, Teddy M dan Prijono, Agus. 2005. Konsep dan Implementasi Struktur Data. Informatika : Bandung. 4. Niswar, M., Ilham, A., Zainuddin, Z., Adnan, A., P, A., Warni, E., Aswad, I., & Muslimin, Z. (2021). Sosialisasi Metode Berfikir Komputasional pada Pendidikan Dasar dan Menengah di Lingkup Sulawesi Selatan. URNAL EPAT eknologi erapan ntuk engabdian asyarakat, 4(1), 46-52. https://doi.org/10.25042/jurnal_tepat.v4i1.172 						
	Pendukung:							
	Tuliskan pustaka pendukung jika ada, sebagai pengayaan literasi							
Dosen Pengampu		<ol style="list-style-type: none"> 1. Dr. Ir. Ingrid Nurtanio., MT 2. Dr. Amil Ahmad Ilham., ST., M.IT 3. Adnan., ST., MT., PhD 4. Anugrayani Bustamin., ST., MT 5. Elly Warni, S.T., M.T 						
Matakuliah syarat		Dasar Pemrograman Komputer						
Pekan Ke-	Sub-CPMK (Kemampuan akhir tiap tahapan belajar)			Bentuk Pembelajaran,		Materi Pembelajaran [Pustaka]	Bobot Penilaian (%)	
		Penilaian		Metode Pembelajaran, Penugasan Mahasiswa, [Estimasi Waktu]				
(1)	(2)	(3)	(4)	Luring (offline)	Daring (online)	(7)	(8)	

1	Mahasiswa mampu mengetahui konsep dan pengertian Algoritma	Mengetahui konsep definisi, manfaat, mekanisme dan kontribusi Algoritma	Assignments	Lecture TM: 4 x 50 menit.	Lecture VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Konsep dan definisi algoritma 2. Mekanisme dan Kontribusi Algoritma	5%
2	Mahasiswa mampu memahami Dasar Algoritma CPL 3- SubCPMK 2	Memahami Dasar Algoritma (tipe data, aturan penulisan, proses, instruksi dan aksi)	quizzes	Lecture, case study TM: 4 x 50 menit.	Lecture, case study VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Dasar Algoritma (tipe data, aturan penulisan, proses, instruksi, dan aksi) Pustaka : [1]	5%
3	Mahasiswa mampu mengenali Flowchart CPL 4- SubCPMK 3	Mampu mengenali Flowchart (simbol, variabel, algoritma cabang, dan runtunan)	Assignments	Lecture, small group discussion TM: 4 x 50 menit.	Lecture, small group discussion VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit	Flowchart (Simbol, variabel, algoritma cabang, runtunan) Pustaka : [1]	7%

4	Mahasiswa mampu memahami bentuk pemilihan (alih kontrol dan pengulangan)	Mampu memahami bentuk pemilihan (alih kontrol dan pengulangan)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Pemilihan Kontrol Pengulangan) Pustaka : [1]	(Alir dan 7%
5	Mahasiswa mampu mengimplementasikan prosedur, fungsi dan pemrosesan teks	Mampu mengimplementasikan pemanggilan prosedur, pemanggilan fungsi dan pemrosesan teks	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Pemanggilan Prosedur 2. Pemanggilan Fungsi 3. Pemrosesan Teks Pustaka : [1]	7%
6	Mahasiswa mampu mempraktekkan konsep array	Mampu mempraktekkan konsep array seperti menginisialisasi array, mengisi elemen, mencari	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Menginisialisasi array 2. Mengisi elemen array 3. Mencari nilai maksimum dan minimum	7%

		nilai maksimum dan minimum			/104D4224/inde x.php?id_session =13610	Pustaka : [1][2]	
7	Mahasiswa mampu menetapkan ciri dari teknik pencarian	Mampu menetapkan ciri dari teknik pencarian ● Binary Search ● Sequential Search	Assignments quizzes	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Binary Search 2. Sequential Search Pustaka : [2][3]	7%
8	Evaluasi Tengah Semester / Ujian Tengah Semester						
9	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting ● Bubble Sort ● Insertion Sort	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Bubble Sort 2. Insertion Sort Pustaka : [2][3]	9%
	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting Merge Sort	Assignments	Lecture, practicum TM: 4 x 50	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit.	1. Merge Sort Selection Sort Pustaka : [2][3]	9%

10		Selection Sort		menit.	BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610		
11	Mahasiswa mampu menetapkan ciri dari teknik sorting	Mampu menetapkan ciri dari teknik sorting Quick Sort berserta complexitas Algoritmanya	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Quick Sort 2. Complexitas Algoritma Pustaka : [2][3]	9%
12	Mahasiswa mampu memahami struct dan konsep linked list	Mampu memahami struct dan konsep dan operasi dasar linked list (deklarasi simpul, penambahan simpul pada linked list awal, akhir dan tengah)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Konsep Struct dan implementasinya 2. Deklarasi list 3. Penambahan simpul pada list awal, akhir dan tengah Pustaka : [2][3]	7%

13	Mahasiswa mampu memahami Antrian dan Stack	Mampu memahami Antrian dan Stack dalam List	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	Antrian dan Stack dalam list Pustaka : [2][3]	7%
14	Mahasiswa mampu memahami teknik graph	mampu memahami teknik graph (graph berarah dan tidak berarah menggunakan linked list)	Assignments	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610	1. Graph berarah Graph tidak berarah Pustaka : [2][3]	7%
15	Mahasiswa mampu mengimplementasi konsep Tree	Mampu mengimplementasi konsep Tree menggunakan linked list	Assignments quizzes	Lecture, practicum TM: 4 x 50 menit.	Lecture, practicum VC: 4 x 50 menit. PT: 4 x 60 menit. BM: 4 x 60 menit	Konsep tree menggunakan linked List Pustaka : [2][3]	7%

					https://sikola.unhas.ac.id/courses/104D4224/index.php?id_session=13610		
16	Evaluasi Akhir Semester / Ujian Akhir Semester						

MODUL ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA

MODUL XII

TREE

Pengantar

Modul ini dirancang untuk membantu mahasiswa memahami konsep dasar mengenai *tree*, yang merupakan salah satu struktur data yang sangat penting dalam pemrograman dan algoritma. *Tree* adalah jenis graf khusus yang tidak memiliki siklus dan memastikan seluruh simpul (node) dapat terhubung satu sama lain. Dalam modul ini, mahasiswa akan mempelajari struktur *tree*, termasuk simpul, sisi, dan properti yang membedakan *tree* dengan jenis graf lainnya.

Pada Sub-CPMK kali ini, mahasiswa diharapkan mampu memahami konsep *disjoint set* dan aplikasinya dalam struktur data untuk menangani pengelompokan elemen yang tidak saling beririsan. Kegiatan belajar akan dilakukan melalui diskusi kelompok kecil dan pembelajaran melalui ceramah serta didukung oleh platform pembelajaran daring.

Kegiatan pembelajaran ini akan mencakup,

- Tatap Muka (TM): 4 sesi, masing-masing 50 menit.
- Virtual Classroom (VC): 4 sesi, masing-masing 50 menit.
- Praktikum Terbimbing (PT): 4 sesi, masing-masing 60 menit.
- Bimbingan Mandiri (BM): 4 sesi, masing-masing 60 menit

Sumber pembelajaran yang dapat diakses untuk mendukung pemahaman lebih lanjut tersedia di platform sikola, dengan pustaka referensi yang telah disediakan.

Setelah mengikuti modul ini, mahasiswa diharapkan dapat mencapai tingkat pemahaman yang cukup, setidaknya 80%, sebelum melanjutkan ke materi berikutnya

KEGIATAN BELAJAR 1

TREE

A. Deskripsi Singkat

Pemahaman *tree* dan Disjoint Set Union (DSU) penting karena kedua struktur data ini memainkan peran penting dalam menganalisa komponen graf seperti ingin menghitung banyaknya komponen di graf atau ingin menghitung yang mana suatu node bisa terhubung ke berapa node lainnya.

Dalam kegiatan belajar pertama ini, mahasiswa akan diperkenalkan pada konsep dasar *tree*, khususnya sifat-sifat *tree* yang membedakannya dari graf. *Tree* adalah struktur data yang terdiri dari simpul (node) yang terhubung secara hierarkis tanpa adanya siklus, dimana setiap node memiliki satu *parent* dan bisa memiliki banyak *child*.

Mahasiswa juga akan mempelajari *disjoint set*, yang digunakan untuk mengelompokkan elemen-elemen yang tidak saling beririsan dan sering diterapkan dalam algoritma *union-find* untuk memeriksa apakah dua elemen berada dalam kelompok yang sama.

Teori yang akan disampaikan mencakup aplikasi praktis dari *tree* dan *disjoint set*, misalnya dalam struktur data untuk jaringan atau algoritma pencarian komponen terhubung. Mahasiswa akan diajarkan cara menggambarkan *tree*, serta mempelajari teknik dasar dalam penggabungan dan pencarian elemen dengan menggunakan *disjoint set*. Dengan pemahaman yang baik, mahasiswa diharapkan dapat memanfaatkan *tree* dan *disjoint set* untuk menyelesaikan berbagai masalah dalam pemrograman yang melibatkan hierarki data, pengelompokan elemen, dan pencarian efisien.

B. Relevansi

Materi dalam modul sebelumnya membahas tentang struktur data dasar array yang penting untuk pengelolaan data dalam bentuk yang lebih sederhana dan terstruktur. Pemahaman tentang array menjadi landasan yang kuat saat mahasiswa mulai mempelajari struktur data yang lebih kompleks seperti *tree* dan Disjoint Set Union (DSU). Keduanya memiliki peran yang sangat penting dalam berbagai algoritma dan aplikasi yang melibatkan pengelolaan hierarki, penggabungan kelompok, serta pencarian efisien dalam struktur data.

Disjoint Set Union (DSU), yang sering disebut sebagai Union-Find, adalah struktur data yang digunakan untuk memecah masalah pengelompokan dan mencari komponen-komponen terhubung dalam graf atau himpunan data. DSU sangat penting dalam algoritma-algoritma yang melibatkan penggabungan dan pencarian komponen yang terhubung, seperti dalam algoritma Kruskal untuk menemukan Minimum Spanning Tree (MST).

Dengan pemahaman yang kuat tentang konsep ini, mahasiswa akan lebih siap untuk merancang dan mengimplementasikan solusi pemrograman yang efisien dan terstruktur dengan pendekatan berbasis algoritma yang ada.

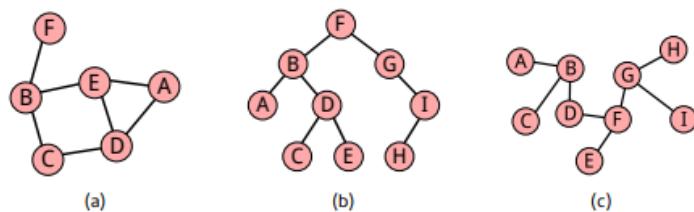
C. Pembelajaran

Capaian pembelajaran yang diharapkan pada kegiatan belajar yang pertama ini adalah peserta kuliah mampu memahami konsep dasar tree sebagai struktur data yang terhubung tanpa cycle, serta dapat menjelaskan hubungan antara jumlah node dan edge dalam tree. Selain itu, peserta kuliah diharapkan mampu menerapkan dan mengimplementasikan struktur data Disjoint Set untuk menyelesaikan masalah keterhubungan dalam graf. Untuk mencapai kompetensi lulusan ini, peserta kuliah diharapkan membaca dan menyelesaikan Kegiatan Belajar 1 ini. Untuk mencapai capaian pembelajaran pada Kegiatan Belajar 1, peserta kuliah harus mengikuti urutan kegiatan pembelajaran yang telah disiapkan dalam modul ini.

1. Uraian

1.1 Konsep Tree

Tree merupakan bentuk khusus dari graf. Seluruh node pada tree terhubung (tidak ada node yang tidak dapat dikunjungi dari node lain) dan tidak terdapat cycle. Cycle merupakan sekumpulan K node unik $[n_0, n_1, \dots, n_{K-1}]$, yang mana $K > 1$, dan setiap elemen ni memiliki tetangga ke $n(i+1) \bmod K$. Banyaknya edge dalam sebuah tree pasti $V - 1$, dengan V adalah banyaknya node .



Gambar (a) bukan tree karena memiliki cycle salah satunya pada note A, E, D. Untuk Gambar (b) dan (c) merupakan tree.

1.2 Disjoint Set

Disjoint Set adalah struktur data yang digunakan untuk mengelompokkan beberapa elemen menjadi sejumlah himpunan yang saling tidak beririsan. Struktur ini sering digunakan dalam operasi seperti Union-Find untuk menangani grup elemen dan memeriksa hubungan antara elemen-elemen tersebut, terutama dalam graf seperti ketika memeriksa apakah dua node berada di komponen yang sama.

a) Inisialisasi Disjoint Set

Pada awalnya, setiap elemen di set sebagai kelompoknya sendiri, yang berarti elemen tersebut menunjuk ke dirinya sendiri sebagai "parent". Dengan kata lain, elemen yang berbeda memiliki perwakilan kelompok yang berbeda pula.

Kode C++ inisialisasi:

```
#include <iostream>
#include <vector>
using namespace std;

vector<int> par;

void initialize(int N) {
    par.resize(N);
    for (int i = 0; i < N; i++) {
        par[i] = i; // Setiap elemen menunjuk ke dirinya sendiri
    }
}
```

b) Operasi Find (FindRepresentative)

Operasi `findRepresentative(x)` mengembalikan elemen perwakilan dari kelompok tempat elemen `x` berada. Perwakilan elemen adalah elemen yang menjadi "root" dari kelompok tersebut. Kita mencari

perwakilan dengan menelusuri setiap parent sampai kita menemukan elemen yang menunjuk pada dirinya sendiri.

Implementasi tanpa path compression:

```
int findRepresentative(int x) {  
    if (par[x] == x) {  
        return x;  
    } else {  
        return findRepresentative(par[x]);  
    }  
}
```

c) Operasi Join (Union)

Operasi join(a, b) menggabungkan dua kelompok yang berisi elemen a dan b dengan cara menjadikan perwakilan kelompok dari a sebagai parent dari perwakilan kelompok b atau sebaliknya. Hal ini akan membuat kedua elemen berada di kelompok yang sama.

```
void join(int a, int b) {  
    int repA = findRepresentative(a);  
    int repB = findRepresentative(b);  
    if (repA != repB) {  
        par[repA] = repB; // Gabungkan kelompok dengan  
        mengubah parent  
    }  
}
```

d) Path Compression untuk Efisiensi

Path Compression digunakan untuk mempercepat operasi findRepresentative(x). Ketika mencari perwakilan kelompok, setiap elemen yang dilalui diperbarui parent-nya langsung ke perwakilan kelompok. Ini memperpendek jalur antara elemen dan

perwakilannya, sehingga pencarian perwakilan berikutnya menjadi lebih cepat.

```
int findRepresentative(int x) {
    if (par[x] != x) {
        par[x] = findRepresentative(par[x]); // Path compression
    }
    return par[x];
}
```

e) Operasi Check

Operasi check(a, b) memeriksa apakah dua elemen berada dalam kelompok yang sama, yaitu jika kedua elemen tersebut memiliki perwakilan yang sama.

```
bool check(int a, int b) {
    return findRepresentative(a) ==
    findRepresentative(b);
}
```

Analisis Kompleksitas

Apabila seluruh parent elemen sudah dikenakan path compression, maka setiap elemen langsung menunjuk ke elemen perwakilan kelompoknya. Artinya, kini fungsi findRepresentative bekerja dalam O(1). Kompleksitas satu kali pemanggilan findRepresentative tidak dapat didefinisikan secara pasti. Perhitungan secara matematis membuktikan bahwa dengan path compression, kompleksitas untuk M operasi findRepresentative dan N - 1 operasi JOIN, dengan $M \geq N$, bekerja dalam $O(M \log N)$.

2. Rangkuman

Tree adalah struktur khusus dari graf di mana semua node terhubung tanpa membentuk cycle, dan jumlah edge-nya selalu $V - 1$, dengan V sebagai jumlah node. Disjoint Set Union (DSU) merupakan teknik yang efisien untuk memeriksa keterhubungan dan mendeteksi cycle dalam graf, termasuk untuk membangun struktur tree. Dengan DSU, setiap node memiliki "parent" yang menunjukkan perwakilan grupnya, memungkinkan kita mengelompokkan node dengan operasi Union dan menemukan root kelompok dengan operasi Find. Dalam konteks tree, DSU memastikan tidak ada cycle saat menambahkan edge antar node, karena dua node yang berada di kelompok yang sama akan memiliki perwakilan yang sama dan tidak perlu dihubungkan kembali.

3. Pustaka

- a. Aji, Alham Fikri dan Gozali, William. *Pemrograman Kompetitif Dasar*. ISBN 978-602-6598-89-9.
- b. Rafif, Muhammad Atpur. (2023). *Aplikasi Graf dan Disjoint Set Union dalam Keterhubungan Bangunan Kampus Ganesha ITB*.

D. Penilaian Kegiatan Belajar

1. Penilaian

1.1 Latihan

1) Perkenalan Disjoint Set

(<https://tlx.toki.id/courses/competitive-1/chapters/10/problems/A>)

Deskripsi

Pak Dengklek memiliki N kandang bebek, dinomori 1 hingga N . Pak Dengklek berencana membangun jalan setapak yang menghubungkan kandang-kandang tersebut.

Pak Dengklek meminta bantuan Anda untuk menjalankan Q operasi, yang masing-masing berupa salah satu dari:

- $\text{join}(a,b)$
- Pak Dengklek membangun jalan antara kandang a dengan kandang b.
- Kini, Pak Dengklek dapat berpindah dari kandang a ke kandang b (dan sebaliknya) dengan jalan ini.
- $\text{check}(a,b)$
- Pak Dengklek ingin mengetahui, apakah kandang a dan kandang b terhubung?

Dengan kata lain, apakah Pak Dengklek dapat berpindah dari kandang a ke kandang b (dan sebaliknya) dengan serangkaian jalan?

Format Masukan

Baris pertama berisi bilangan bulat N dan Q.

Q baris berikutnya masing-masing berisi sebuah operasi, dengan format:

- 1 a b: artinya lakukan $\text{join}(a,b)$.
- 2 a b: artinya lakukan $\text{check}(a,b)$.

Format Keluaran

Untuk setiap operasi $\text{check}(a,b)$:

- Cetak Y apabila kandang a dan b terhubung
- Cetak T apabila kandang a dan b tidak terhubung.

Contoh Masukan

5 7

1 1 4

2 1 2

1 1 2

2 1 2

1 3 5

1 2 3

2 1 5

Contoh Keluaran

T

Y

Y

Batasan

- $1 \leq N, Q \leq 100000$
- $1 \leq a, b \leq N$

1.2 Tantangan

1) Cutting a graph

(<https://codeforces.com/edu/course/2/lesson/7/1/practice/contest/289390/problem/D>)

time limit per test - 2 seconds

memory limit per test - 256 megabytes

There is an undirected graph and a sequence of operations of two types in the following format:

- cut $u v$ — remove edge $u - v$ from the graph;
- ask $u v$ — check whether vertices u and v are in the same connected component.

After all the operations are applied, the graph contains no edges. Please, find the result of each operation of type ask.

Input

First line of input consists of three integers n , m and k ($1 \leq n \leq 50\,000$, $0 \leq m \leq 100\,000$, $m \leq k \leq 150\,000$) — the number of vertices in the graph, the number of edges and the number of operations, respectively.

Each of next m lines consists of two integers u_i and v_i ($1 \leq u_i, v_i \leq n$) — ends of edge i . Vertices are numbered from 1, graph has no loops and multiple edges.

Each of the next k lines describes an operation in the following format:

- "cut u v " ($1 \leq u, v \leq n$) — remove an edge between vertices u and v
- "ask u v " ($1 \leq u, v \leq n$) — check whether vertices u and v are in the same component

Each edge is mentioned in operations of type cut once.

Output

For each of operation of type ask output "YES", if two given vertices are in the same component, and "NO" — otherwise.

The order of the answers should correspond to the order of operations of type ask in input.

Example:

Input

3 3 7

1 2

2 3

3 1

ask 3 3

cut 1 2

ask 1 2

cut 1 3

ask 2 1

cut 2 3

ask 3 1

Output

YES

YES

NO

NO

2. Umpang Balik dan Tindak Lanjut

1) Source Code Latihan (Perkenalan Disjoint Set)

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long

vector<ll> parent(1e5+5);
vector<ll> r(1e5+5, 0);
ll N, Q, type, a, b;

ll get(ll x) {
    if (parent[x] == x)
        return x;
    return get(parent[x]);
}

void unite(ll x, ll y) {
    x = get(x);
    y = get(y);

    if (x!= y) {
        if (r[x] < r[y]) {
            parent[x] = y;
        } else if (r[x] > r[y]) {
            parent[y] = x;
        } else {
            parent[y] = x;
            r[x]++;
        }
    }
}
```

```
int main() {
    cin >> N >> Q;

    for (int i = 1; i <= N; i++) {
        parent[i] = i;
    }

    for (ll i = 0; i < Q; i++) {
        cin >> type >> a >> b;

        if (type == 2) {
            if (get(a) == get(b)) {
                cout << "Y" << endl;
            } else {
                cout << "T" << endl;
            }
        } else {
            unite(a, b);
        }
    }
}
```