Algoritma dan Pemrograman

Buku Ajar Berbasis Outcome-Based Education (OBE)

Mata Kuliah: Algoritma dan Pemrograman

Kode: TIF-101

Bahasa Pemrograman: C

Program Studi Teknik Informatika

Fakultas Teknik

Universitas Lorem Ipsum

2026

Kata Pengantar

Puji syukur kehadirat Tuhan Yang Maha Esa atas terselesaikannya buku ajar *Algoritma dan Pemrograman* ini. Buku ini disusun dengan pendekatan **Outcome-Based Education (OBE)**, yang berfokus pada pencapaian kompetensi terukur sesuai dengan Capaian Pembelajaran Lulusan (CPL) dan Capaian Pembelajaran Mata Kuliah (CPMK).

Algoritma dan Pemrograman merupakan mata kuliah fundamental yang harus dikuasai oleh setiap mahasiswa Teknik Informatika. Mata kuliah ini tidak hanya mengajarkan cara menulis kode dalam bahasa C, tetapi juga cara berpikir algoritmik dalam merancang solusi permasalahan komputasi secara sistematis.

Buku ini dirancang untuk mendukung pembelajaran mahasiswa semester 1 dengan pendekatan student-centered learning. Setiap bab dilengkapi dengan:

- Sub-CPMK yang jelas dan terukur
- Materi pokok dengan contoh kode C yang lengkap
- Aktivitas pembelajaran yang mendorong eksplorasi mandiri
- Latihan dan refleksi untuk penguatan pemahaman
- Asesmen untuk mengukur pencapaian kompetensi
- Checklist kompetensi untuk self-assessment

Kami berharap buku ini dapat menjadi panduan yang efektif dalam perjalanan pembelajaran Anda menguasai Algoritma dan Pemrograman.

Penyusun
Dr. Lorem Ipsum, M.Kom.

Cara Menggunakan Buku Ini

Buku ajar ini dirancang dengan pendekatan OBE untuk memaksimalkan pencapaian pembelajaran Anda. Berikut panduan penggunaan buku ini:

Struktur Buku

Bab I: Pendahuluan dan Orientasi
Memperkenalkan tujuan buku, keterkaitan dengan RPS, dan konteks kurikulum OBE.

Bab II: Landasan Teori
Menyajikan fondasi teoretis algoritma dan pemrograman yang menjadi basis pembelajaran seluruh bab berikutnya.

Bab III-XVI: Unit Materi Inti
Setiap bab mencakup satu topik utama Algoritma dan Pemrograman (flowchart, pseudocode, variabel, percabangan, perulangan, fungsi, array, string, struct, pengurutan, pencarian) dengan struktur lengkap: Sub-CPMK, materi, aktivitas, latihan, asesmen, dan checklist.

Bab XVII: Evaluasi dan Integrasi
Berisi asesmen komprehensif dan panduan refleksi untuk mengukur pencapaian kompetensi secara menyeluruh.

Lampiran
Menyediakan rubrik penilaian, template laporan, modul Praktikum 1, glosarium istilah algoritma dan C, serta referensi tambahan.

Komponen dalam Setiap Bab

1. **Sub-CPMK**: Baca dengan seksama untuk memahami kompetensi yang harus dicapai
2. **Materi Pokok**: Pelajari dengan cermat, jalankan semua contoh kode
3. **Aktivitas Pembelajaran**: Lakukan secara mandiri atau berkelompok
4. **Latihan**: Kerjakan untuk menguji pemahaman Anda
5. **Asesmen**: Gunakan untuk mengukur pencapaian Sub-CPMK
6. **Checklist**: Centang setelah yakin menguasai setiap indikator

Tips Belajar Efektif

- Jangan hanya membaca, praktikkan semua contoh kode C
- Gunakan compiler C (GCC, MinGW, atau IDE seperti Code::Blocks, Dev-C++) untuk eksperimen
- Kerjakan latihan sebelum melihat solusi
- Diskusikan konsep yang sulit dengan teman atau dosen
- Manfaatkan checklist untuk self-assessment berkala
- Kerjakan proyek mini untuk mengintegrasikan konsep yang dipelajari

Identitas Mata Kuliah

Nama Program Studi	: Teknik Informatika
Nama Mata Kuliah	: Algoritma dan Pemrograman
Kode Mata Kuliah	: TIF-101
Semester	: 1 (Satu)
SKS / Bobot Kredit	: 3 SKS (2 Teori, 1 Praktikum)
Bahasa Pemrograman	: C
Dosen Pengampu	: Dr. Lorem Ipsum, M.Kom.
Tanggal Penyusunan	: 12 Februari 2026

Capaian Pembelajaran Lulusan (CPL)

CPL yang dibebankan pada mata kuliah ini mencakup kompetensi lulusan dalam aspek pengetahuan, keterampilan, dan sikap:

1. **CPL-1 (Pengetahuan):** Menguasai konsep teoretis bidang pengetahuan tertentu secara umum dan konsep teoretis bagian khusus dalam bidang pengetahuan tersebut secara mendalam, serta mampu memformulasikan penyelesaian masalah prosedural.
2. **CPL-2 (Keterampilan Umum):** Mampu menerapkan pemikiran logis, kritis, sistematis, dan inovatif dalam konteks pengembangan atau implementasi ilmu pengetahuan dan teknologi yang memperhatikan dan menerapkan nilai humaniora.
3. **CPL-3 (Keterampilan Khusus):** Mampu merancang dan mengimplementasikan algoritma serta program komputer sederhana menggunakan bahasa pemrograman prosedural untuk menyelesaikan permasalahan komputasi.
4. **CPL-4 (Sikap):** Menunjukkan sikap bertanggung jawab atas pekerjaan di bidang keahliannya secara mandiri dan mampu bekerja sama dalam tim.

Capaian Pembelajaran Mata Kuliah (CPMK)

Kemampuan atau kompetensi spesifik yang diharapkan mahasiswa kuasai setelah menyelesaikan mata kuliah:

1. **CPMK-1:** Mahasiswa mampu memahami dan menjelaskan konsep algoritma, flowchart, dan pseudocode beserta karakteristiknya.

2. **CPMK-2:** Mahasiswa mampu merancang algoritma untuk menyelesaikan masalah sederhana menggunakan flowchart dan pseudocode.

3. **CPMK-3:** Mahasiswa mampu mengimplementasikan algoritma dalam bahasa C meliputi variabel, tipe data, I/O, operator, percabangan, dan perulangan.

4. **CPMK-4:** Mahasiswa mampu menggunakan fungsi/prosedur, array, dan struktur data dasar dalam bahasa C.

5. **CPMK-5:** Mahasiswa mampu menerapkan algoritma pengurutan dan pencarian sederhana serta mengimplementasikannya dalam program.

Matriks Keterkaitan CPL dan CPMK

CPL	**CPMK**	**Kontribusi**	**Keterangan**
CPL-1	CPMK-1	Tinggi	Penguasaan konsep algoritma dan pemrograman
CPL-1	CPMK-2	Tinggi	Kemampuan merancang algoritma
CPL-2	CPMK-2, CPMK-3	Tinggi	Penerapan pemikiran sistematis dalam pemrograman
CPL-2	CPMK-4, CPMK-5	Sedang	Pemikiran kritis dalam implementasi
CPL-3	CPMK-2, CPMK-3	Tinggi	Perancangan dan implementasi program
CPL-3	CPMK-4, CPMK-5	Tinggi	Penggunaan struktur data dan algoritma
CPL-4	CPMK-3, CPMK-4	Sedang	Tanggung jawab dalam implementasi
CPL-4	CPMK-5	Sedang	Kerja sama dalam proyek

Tabel 1: Matriks Keterkaitan CPL dan CPMK

Daftar Isi

Daftar Gambar

Daftar Tabel

Daftar Kode Program

Bab 1

Pendahuluan dan Orientasi Buku

1.1 Tujuan Buku Ajar

Buku ajar ini dirancang sebagai panduan komprehensif untuk menguasai Algoritma dan Pemrograman menggunakan bahasa C. Fokus utama buku ini adalah pada pemahaman konsep algoritma, kemampuan merancang solusi, dan implementasi program komputer secara prosedural. Tujuan spesifik buku ini adalah memberikan landasan yang kokoh bagi mahasiswa Teknik Informatika sebelum mempelajari mata kuliah pemrograman lanjutan.

Setelah mempelajari buku ini secara menyeluruh, mahasiswa diharapkan mampu memahami dan menjelaskan konsep algoritma, flowchart, serta pseudocode beserta karakteristiknya. Mahasiswa juga diharapkan mampu merancang algoritma untuk menyelesaikan masalah sederhana menggunakan flowchart dan pseudocode. Selain itu, mahasiswa harus mampu mengimplementasikan algoritma dalam bahasa C meliputi variabel, tipe data, I/O, operator, percabangan, dan perulangan.

Mahasiswa diharapkan mampu menggunakan fungsi, prosedur, array, dan struktur data dasar dalam bahasa C. Terakhir, mahasiswa harus mampu menerapkan algoritma pengurutan dan pencarian sederhana serta mengimplementasikannya dalam program. Semua tujuan tersebut dirancang untuk memfasilitasi pencapaian CPL dan CPMK yang telah ditetapkan dalam RPS mata kuliah Algoritma dan Pemrograman.

1.2 Keterkaitan Buku Ajar dengan RPS Berbasis OBE

Buku ajar ini dirancang selaras dengan Rencana Pembelajaran Semester (RPS) mata kuliah Algoritma dan Pemrograman yang berbasis OBE. Keterkaitan ini diwujudkan melalui pemetaan eksplisit setiap bab ke Sub-CPMK yang berkontribusi pada pencapaian CPMK dan CPL program studi. Struktur ini memastikan bahwa materi pembelajaran fokus pada pencapaian kompetensi terukur sesuai dengan matriks 16 pertemuan dalam silabus.

1.2.1 Alignment dengan CPL dan CPMK

Setiap bab dalam buku ini dipetakan secara eksplisit ke Sub-CPMK yang berkontribusi pada pencapaian CPMK dan CPL. Materi pembelajaran dirancang untuk mendukung pengembangan keterampilan merancang algoritma dan mengimplementasikan program dalam bahasa C. Asesmen dalam setiap bab mengukur pencapaian kompetensi secara objektif sesuai indikator yang telah ditetapkan.

1.2.2 Integrasi Metode Pembelajaran

Buku ini mengintegrasikan berbagai metode pembelajaran yang tercantum dalam RPS, antara lain ceramah interaktif, Problem-Based Learning (PBL), praktikum terbimbing, serta diskusi dan studi kasus. Aktivitas pembelajaran mendorong mahasiswa untuk mempraktikkan coding di laboratorium dan menyelesaikan tugas individu maupun proyek kecil. Komponen asesmen sejalan dengan sistem penilaian RPS meliputi tugas individu, kuis, praktikum, UTS, tugas proyek, dan UAS.

1.3 Petunjuk Penggunaan Buku Ajar

1.3.1 Untuk Mahasiswa

Sebelum perkuliahan, bacalah Sub-CPMK di awal bab untuk memahami target pembelajaran yang harus dicapai. Pelajari materi pokok dengan seksama dan jalankan semua contoh kode C yang diberikan menggunakan compiler seperti GCC atau MinGW. Catat pertanyaan atau konsep yang belum dipahami untuk didiskusikan di kelas.

Selama perkuliahan, diskusikan konsep yang sulit dengan dosen dan teman secara aktif. Kerjakan aktivitas pembelajaran dan praktikum dengan sungguh-sungguh di laboratorium. Manfaatkan kesempatan untuk bertanya dan berpartisipasi dalam diskusi kelompok. Setelah perkuliahan, kerjakan latihan dan refleksi, lakukan asesmen mandiri, serta centang checklist kompetensi yang telah dikuasai. Kerjakan proyek mini untuk memperdalam pemahaman dan mengintegrasikan konsep yang dipelajari.

1.3.2 Untuk Dosen

Buku ini dapat digunakan sebagai bahan ajar utama untuk perkuliahan Algoritma dan Pemrograman semester 1. Buku ini menyediakan sumber latihan dan tugas yang selaras dengan CPMK serta panduan untuk merancang aktivitas pembelajaran. Dosen dapat memanfaatkan rubrik asesmen dalam buku untuk mengukur pencapaian CPMK mahasiswa secara konsisten.

1.4 Konteks Kurikulum OBE

1.4.1 Apa itu Outcome-Based Education?

Outcome-Based Education (OBE) adalah pendekatan pembelajaran yang berfokus pada pencapaian hasil (outcomes) yang terukur [14]. Dalam OBE, proses pembelajaran dirancang secara sistematis untuk memastikan mahasiswa mencapai kompetensi yang telah ditetapkan. Prinsip utama OBE meliputi clarity of focus, designing down dari outcomes yang diinginkan, high expectations, dan expanded opportunity bagi semua mahasiswa.

1.4.2 Implementasi OBE dalam Buku Ini

Buku ini mengimplementasikan OBE melalui Sub-CPMK eksplisit di setiap bab, materi yang disusun dari dasar ke lanjut secara sistematis, serta aktivitas beragam berupa latihan, studi kasus, dan praktikum. Asesmen terukur dengan rubrik penilaian yang jelas disertai checklist untuk self-assessment memungkinkan mahasiswa memantau pencapaian kompetensi secara berkelanjutan. Implementasi tersebut dirancang agar pembelajaran di tingkat mikro (per bab) mendukung pencapaian kompetensi di tingkat makro (lulusan program studi).

1.5 Peta Konsep Algoritma dan Pemrograman

Mata kuliah Algoritma dan Pemrograman mencakup 17 bab yang saling terkait secara berurutan. Tabel berikut memetakan bab buku ajar ke 16 pertemuan dalam RPS:

Bab II menyajikan landasan teori dan konsep dasar algoritma serta pengantar bahasa C. Bab III dan IV membahas pengenalan algoritma, flowchart, dan pseudocode sebagai fondasi merancang solusi. Bab V dan VI membahas variabel, tipe data, I/O, serta operator dalam bahasa C.

Bab VII dan VIII membahas struktur percabangan (if-else dan switch-case) untuk pengambilan keputusan dalam program. Bab IX dan X membahas struktur perulangan (for, while, do-while) serta perulangan bersarang. Bab XI membahas fungsi dan prosedur untuk modularisasi program. Bab XII, XIII, dan XIV membahas array satu dimensi, array dua dimensi, serta string dalam C.

Bab XV membahas struct dan tipe data bentukan untuk merepresentasikan data kompleks. Bab XVI membahas algoritma pengurutan dan pencarian sebagai penerapan konsep yang telah dipelajari. Bab XVII berisi evaluasi, refleksi, dan integrasi kompetensi untuk mengukur pencapaian pembelajaran secara menyeluruh. Alur pembelajaran dirancang secara progresif dari konsep dasar hingga penerapan algoritma pada masalah nyata.

Rangkuman

Bab ini memperkenalkan tujuan buku ajar, keterkaitan dengan RPS berbasis OBE, petunjuk penggunaan, dan konteks kurikulum OBE. Pemahaman yang baik tentang struktur

Pert.	**Materi Silabus**	**Bab Buku Ajar**
1	Pengenalan algoritma, flowchart, pseudocode	Bab II, III
2	Variabel, tipe data, I/O, operator	Bab V, VI
3	Percabangan if-else	Bab VII
4	Percabangan switch-case	Bab VIII
5	Perulangan for, while, do-while	Bab IX
6	Perulangan bersarang	Bab X
7	Praktikum 1 (dasar pemrograman)	Aktivitas Bab III–VII
8	Fungsi dan prosedur	Bab XI
9	UTS	–
10	Array satu dimensi	Bab XII
11	Array dua dimensi, matriks	Bab XIII
12	String dan manipulasi teks	Bab XIV
13	Struct/record, pengurutan dan pencarian	Bab XV, XVI
14	Algoritma pengurutan dan pencarian lanjutan	Bab XVI
15	Integrasi, proyek, review materi	Bab XVII
16	UAS	Bab XVII

Tabel 1.1: Pemetaan Bab Buku Ajar ke 16 Pertemuan RPS

dan pendekatan buku ini akan membantu Anda memaksimalkan pembelajaran Algoritma dan Pemrograman.

Poin Kunci:

- Buku ini dirancang dengan pendekatan OBE yang fokus pada pencapaian kompetensi terukur
- Setiap bab dipetakan ke Sub-CPMK yang berkontribusi pada CPL
- Gunakan komponen OBE (Sub-CPMK, aktivitas, latihan, asesmen, checklist) secara optimal
- Pembelajaran Algoritma dan Pemrograman tersusun sistematis dari flowchart hingga algoritma pengurutan dan pencarian

Bab 2

Landasan Teori dan Konsep Dasar Algoritma

Sub-CPMK yang Dicakup dalam Bab Ini:

- Sub-CPMK 1.1: Menjelaskan definisi dan karakteristik algoritma (input, output, definiteness, finiteness, effectiveness, determinism)

Bab ini juga menyajikan materi pengantar pemrograman prosedural dan bahasa C sebagai fondasi untuk CPMK-3.

2.1 Sejarah dan Konsep Algoritma

Kata *algoritma* berasal dari nama matematikawan Persia abad ke-9, Abu Abdullah Muhammad bin Musa al-Khwarizmi (780-850 M). Al-Khwarizmi adalah ilmuwan di House of Wisdom (Bayt al-Hikmah) di Baghdad yang menulis kitab *Al-Jabr wa'l-Muqabala* (Buku tentang Penyempurnaan dan Penyeimbangan) pada tahun 825 M. Kitab ini memperkenalkan metode sistematis untuk menyelesaikan persamaan aljabar linear dan kuadrat. Namanya dilatinkan menjadi "Algoritmi" oleh para penerjemah Eropa dan kemudian berkembang menjadi "algorithm" dalam bahasa Inggris.

2.1.1 Kontribusi Ilmiah al-Khwarizmi

Al-Khwarizmi memberikan kontribusi fundamental dalam beberapa bidang:

- **Matematika:** Memperkenalkan sistem desimal posisional dan konsep nol dari India ke dunia Islam dan Eropa
- **Aljabar:** Mengembangkan metode sistematis untuk menyelesaikan persamaan dengan langkah-langkah terstruktur

- **Astronomi:** Membuat tabel astronomi yang akurat untuk navigasi dan penentuan waktu shalat
- **Geografi:** Memperbaiki karya Ptolemy dan membuat peta dunia yang lebih akurat

Kontribusinya sangat fundamental dalam matematika komputasi karena mengenalkan pendekatan langkah-demi-langkah (step-by-step) dalam pemecahan masalah, yang menjadi cikal bakal konsep algoritma modern.

Dalam ilmu komputer modern, **algoritma** didefinisikan sebagai urutan langkah-langkah logis, terdefinisi dengan jelas, dan terbatas untuk menyelesaikan suatu masalah atau mencapai tujuan tertentu [6]. Algoritma menjadi fondasi fundamental dalam pemrograman karena program komputer pada hakikatnya adalah implementasi algoritma dalam bahasa pemrograman. Setiap program yang kita tulis adalah representasi dari algoritma yang dirancang untuk menyelesaikan masalah spesifik.

2.1.2 Perkembangan Historis Algoritma

Sejarah algoritma tidak terlepas dari perkembangan pemrograman komputer:

- **Era Pra-Komputer (1940-an):** Konsep algoritma sudah ada dalam matematika, namun implementasi masih manual
- **Era Machine Language (1950-an):** Pemrograman dilakukan dengan bahasa mesin dan assembly, algoritma sangat sederhana
- **Era Bahasa Tingkat Tinggi (1960-1970-an):** FORTRAN, COBOL, C, dan Pascal berkembang dengan pendekatan terstruktur
- **Era Modern (1980-an-sekarang):** Algoritma kompleks untuk AI, machine learning, big data

Bahasa C yang dikembangkan oleh Dennis Ritchie di Bell Labs pada 1972 menjadi bahasa prosedural yang sangat berpengaruh dan menjadi fondasi bagi banyak bahasa pemrograman modern seperti C++, Java, dan Python.

2.1.3 Pentingnya Algoritma dalam Pemrograman

Pemahaman algoritma penting karena beberapa alasan:

1. **Efisiensi:** Algoritma yang baik menyelesaikan masalah dengan penggunaan sumber daya minimal
2. **Kebenaran:** Memastikan solusi yang dihasilkan selalu benar untuk semua kasus input
3. **Skalabilitas:** Algoritma yang efisien dapat menangani data dalam jumlah besar
4. **Pemeliharaan:** Algoritma yang jelas mudah dipahami dan dimodifikasi

Dalam mata kuliah Algoritma dan Pemrograman, mahasiswa belajar merancang algoritma menggunakan flowchart dan pseudocode sebelum mengimplementasikannya dalam bahasa C. Pendekatan ini memastikan mahasiswa memahami logika solusi sebelum terjun ke implementasi teknis.

2.2 Pemrograman Prosedural dan Paradigma Lain

Pemrograman prosedural adalah paradigma yang berfokus pada prosedur atau fungsi yang beroperasi pada data [15]. Program terdiri dari serangkaian instruksi yang dieksekusi secara berurutan, dengan data dan fungsi dapat dipisahkan. Bahasa C adalah contoh utama bahasa pemrograman prosedural yang masih digunakan luas hingga saat ini. Mata kuliah Algoritma dan Pemrograman menggunakan pendekatan prosedural karena memungkinkan fokus pada logika algoritmik tanpa kompleksitas paradigma lain.

Selain prosedural, terdapat paradigma pemrograman lain seperti pemrograman berorientasi objek (OOP) yang berfokus pada objek dan class, serta pemrograman fungsional yang menekankan evaluasi fungsi. Pemrograman prosedural cocok untuk program yang relatif kecil, tugas komputasi langsung, dan pembelajar pemrograman pemula. Pemahaman prosedural menjadi fondasi penting sebelum mempelajari paradigma yang lebih kompleks seperti OOP di semester berikutnya.

Paradigma	**Karakteristik Utama**
Prosedural	Fungsi dan data terpisah, alur top-down, bahasa C, Pascal
Berorientasi Objek	Class dan objek, enkapsulasi, bahasa Java, C++, Python
Fungsional	Fungsi sebagai nilai, bahasa Haskell, Lisp

Tabel 2.1: Perbandingan Paradigma Pemrograman

2.3 Pengantar Bahasa C

Bahasa C dikembangkan oleh Dennis Ritchie di Bell Labs pada tahun 1972 untuk sistem operasi UNIX. C menjadi salah satu bahasa pemrograman paling berpengaruh karena efisiensinya, portabilitasnya, dan kemampuannya melakukan manipulasi level rendah [3]. Banyak bahasa pemrograman modern seperti C++, Java, dan Python terpengaruh oleh sintaks dan konsep C. Dalam mata kuliah ini, C dipilih karena kesederhanaannya dan kesesuaiannya untuk mempelajari dasar-dasar algoritma dan pemrograman prosedural.

Program C terdiri dari fungsi-fungsi, dengan fungsi `main()` sebagai titik masuk eksekusi. C mendukung tipe data dasar seperti `int`, `float`, `char`, dan `double`, serta struktur kontrol seperti percabangan dan perulangan. Untuk mengompilasi program C, dibutuhkan compiler seperti GCC (GNU Compiler Collection) atau MinGW di Windows. Setelah di-

kompilasi, program C menghasilkan file executable yang dapat dijalankan langsung oleh sistem operasi.

Contoh

Contoh program C minimal yang menampilkan teks ke layar:

```c
#include <stdio.h>

int main() {
    printf("Hello, World!\n");
    return 0;
}
```

Kode Program 2.1: Program Hello World dalam C

Baris `#include <stdio.h>` menyertakan header untuk fungsi input-output. Fungsi `printf` menampilkan teks ke layar, dan `return 0` mengindikasikan program berakhir sukses.

2.4 Karakteristik Algoritma

Berdasarkan standar ilmu komputer modern dan referensi dari GeeksforGeeks, sebuah algoritma harus memiliki enam karakteristik fundamental untuk dapat dikategorikan sebagai algoritma yang baik dan benar [6]. Karakteristik ini menjadi acuan dalam mengevaluasi kualitas algoritma yang dirancang.

2.4.1 Enam Karakteristik Utama Algoritma

1. **Input (Masukan)**
   - Algoritma harus memiliki nol atau lebih input yang terdefinisi dengan jelas
   - Input harus spesifik dan dapat diukur
   - Contoh: algoritma pengurutan memerlukan array sebagai input
2. **Output (Keluaran)**
   - Algoritma harus menghasilkan minimal satu output
   - Output harus sesuai dengan tujuan algoritma
   - Contoh: array yang sudah terurut, nilai maksimum, hasil perhitungan
3. **Definiteness (Kejelasan)**
   - Setiap langkah harus didefinisikan dengan jelas dan tidak ambigu
   - Tidak boleh ada interpretasi ganda untuk setiap instruksi
   - Contoh yang salah: "tambahkan sedikit garam" → tidak jelas
   - Contoh yang benar: "tambahkan 1 sendok teh garam" → jelas

4. **Finiteness (Keterbatasan)**
   - Algoritma harus berakhir setelah jumlah langkah terbatas
   - Tidak boleh mengandung infinite loop tanpa kondisi berhenti
   - Setiap eksekusi harus selesai dalam waktu wajar
5. **Effectiveness (Efektivitas)**
   - Setiap langkah harus dapat dilakukan dengan sumber daya yang tersedia
   - Instruksi harus dasar dan dapat dieksekusi dalam waktu terbatas
   - Tidak boleh memerlukan operasi yang tidak mungkin dilakukan
6. **Determinism (Deterministik)**
   - Untuk input yang sama, algoritma harus selalu menghasilkan output yang sama
   - Tidak ada elemen random atau kebetulan dalam proses
   - Hasil dapat diprediksi dan direproduksi

2.4.2 Tabel Perbandingan Karakteristik

Karakteristik	**Deskripsi**	**Contoh Penerapan**
Input	Data masukan yang diperlukan	Array angka untuk diurutkan
Output	Hasil yang dihasilkan	Array yang sudah terurut
Definiteness	Langkah jelas tidak ambigu	"Tukar jika A > B"
Finiteness	Berakhir dalam waktu terbatas	Loop berhenti setelah n-1 iterasi
Effectiveness	Langkah dapat dilakukan	Operasi perbandingan dan pertukaran
Determinism	Hasil konsisten	Input [3,1,2] selalu menghasilkan [1,2,3]

Tabel 2.2: Enam Karakteristik Algoritma yang Baik

Konsep Penting

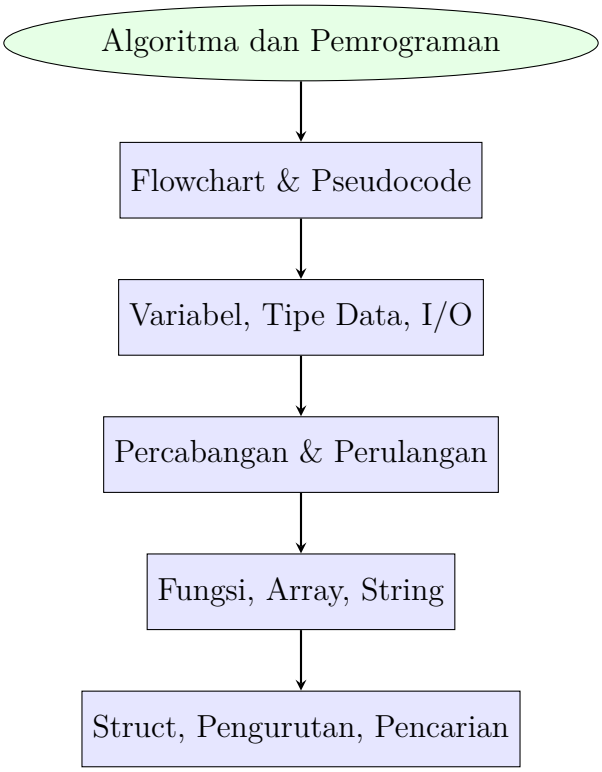
Ringkasan enam karakteristik algoritma:

1. **Input**: Algoritma menerima masukan yang terdefinisi (nol atau lebih)
2. **Output**: Algoritma menghasilkan minimal satu keluaran
3. **Definiteness**: Setiap langkah jelas dan tidak ambigu
4. **Finiteness**: Algoritma berakhir dalam langkah terbatas
5. **Effectiveness**: Setiap langkah dapat dilaksanakan secara praktis
6. **Determinism**: Input yang sama selalu menghasilkan output yang sama

2.5 Peta Konsep dan Contoh Algoritma

2.5.1 Peta Konsep Algoritma dan Pemrograman

Peta konsep mata kuliah Algoritma dan Pemrograman menunjukkan alur pembelajaran dari fondasi hingga penerapan. Dimulai dari pemahaman konsep algoritma dan karakteristiknya, mahasiswa belajar merancang solusi menggunakan flowchart dan pseudocode. Selanjutnya, mahasiswa mempelajari elemen dasar bahasa C meliputi variabel, tipe data, I/O, dan operator.



Struktur percabangan (if-else, switch) dan perulangan (for, while, do-while) memungkinkan pengambilan keputusan dan pengulangan operasi. Fungsi dan prosedur mendukung modularisasi program. Array dan string digunakan untuk mengelola kumpulan data. Struct memungkinkan pembuatan tipe data bentukan. Algoritma pengurutan dan pencarian menerapkan semua konsep yang telah dipelajari untuk menyelesaikan masalah komputasi nyata.

2.5.2 Contoh Algoritma Lengkap

Algoritma Menemukan Nilai Maksimum dalam Array:

1. **Input:** Array A dengan n elemen bilangan bulat
2. **Proses:**
   (a) Inisialisasi: maks $\leftarrow$ A[0]

(b) Untuk i ← 1 hingga n-1:

- Jika A[i] > maks maka maks ← A[i]

3. **Output:** Nilai maksimum (maks)

Analisis 6 Karakteristik:

- **Input:** Array A dengan n elemen (terdefinisi jelas)
- **Output:** Satu nilai maksimum (spesifik)
- **Definiteness:** Langkah-langkah jelas (inisialisasi, loop, perbandingan, assignment)
- **Finiteness:** Loop berakhir setelah n-1 iterasi
- **Effectiveness:** Operasi perbandingan dan assignment dapat dilakukan
- **Determinism:** Input [3,1,2,5,4] selalu menghasilkan 5

Implementasi dalam Bahasa C:

```c
#include <stdio.h>

int findMaximum(int arr[], int n) {
    int maks = arr[0];  // Inisialisasi

    for (int i = 1; i < n; i++) {
        if (arr[i] > maks) {
            maks = arr[i];  // Update jika ditemukan nilai
                lebih besar
        }
    }

    return maks;  // Output
}

int main() {
    int data[] = {3, 1, 2, 5, 4};
    int n = sizeof(data) / sizeof(data[0]);

    printf("Nilai maksimum: %d\n", findMaximum(data, n));
    return 0;
}
```

Kode Program 2.2: Implementasi Algoritma Nilai Maksimum

Contoh ini menunjukkan bagaimana algoritma yang dirancang dengan baik memenuhi semua karakteristik fundamental dan dapat diimplementasikan dalam bahasa C dengan hasil yang dapat diprediksi.

Aktivitas Pembelajaran

1. **Diskusi Kelompok**: Identifikasi 5 contoh algoritma dalam kehidupan sehari-hari (misalnya: resep masakan, petunjuk penggunaan ATM) dan analisis apakah memenuhi enam karakteristik algoritma.

2. **Analisis Kode**: Jalankan program Hello World dalam C, modifikasi untuk menampilkan nama Anda, lalu identifikasi fungsi `printf` dan `main`.

3. **Perbandingan Paradigma**: Cari informasi tentang perbedaan pemrograman prosedural dan OOP. Buat ringkasan singkat kapan masing-masing cocok digunakan.

4. **Instalasi Compiler**: Install GCC atau MinGW di komputer Anda, lalu kompilasi dan jalankan program C sederhana.

Latihan dan Refleksi

1. Jelaskan definisi algoritma menurut pemahaman Anda! Berikan dua contoh algoritma dari kehidupan sehari-hari.

2. Sebutkan dan jelaskan enam karakteristik algoritma (input, output, definiteness, finiteness, effectiveness, determinism)!

3. Mengapa bahasa C cocok untuk mempelajari dasar-dasar algoritma dan pemrograman?

4. Jelaskan perbedaan utama antara pemrograman prosedural dan pemrograman berorientasi objek!

5. Buat program C minimal yang menampilkan nama dan NIM Anda. Jelaskan fungsi setiap baris kode.

6. **Refleksi**: Bagaimana pemahaman Anda tentang algoritma sebelum dan sesudah mempelajari bab ini? Konsep mana yang paling sulit dipahami?

Asesmen (Evaluasi Kinerja)

Instrumen Penilaian untuk Sub-CPMK 1.1

A. Pilihan Ganda

1. Manakah yang BUKAN merupakan karakteristik algoritma?

   (a) Input
   (b) Output
   (c) Abstraksi
   (d) Finiteness

2. Bahasa C dikembangkan oleh:
   (a) James Gosling
   (b) Dennis Ritchie
   (c) Bjarne Stroustrup
   (d) Guido van Rossum
3. Karakteristik algoritma yang menyatakan bahwa setiap langkah harus jelas dan tidak ambigu disebut:
   (a) Input
   (b) Definiteness
   (c) Effectiveness
   (d) Output

B. Essay

1. Jelaskan dengan kata-kata Anda sendiri apa yang dimaksud dengan algoritma dan berikan 2 contoh algoritma dari kehidupan sehari-hari.
2. Sebutkan dan jelaskan keenam karakteristik algoritma yang baik. Berikan contoh langkah yang memenuhi dan tidak memenuhi karakteristik definiteness dan determinism.

Rubrik Penilaian: Lihat Lampiran A

Checklist Pencapaian Kompetensi

Centang item berikut setelah Anda yakin telah menguasainya:

☐ Saya dapat menjelaskan definisi dan sejarah algoritma

☐ Saya dapat menyebutkan dan menjelaskan enam karakteristik algoritma

☐ Saya memahami perbedaan pemrograman prosedural dan paradigma lain

☐ Saya dapat menulis dan menjalankan program C minimal (Hello World)

☐ Saya memahami peta konsep mata kuliah Algoritma dan Pemrograman

Rangkuman

Bab ini membahas landasan teori Algoritma dan Pemrograman, termasuk sejarah dan konsep algoritma, pemrograman prosedural, pengantar bahasa C, serta karakteristik algoritma.

Poin Kunci:

- Algoritma adalah urutan langkah logis yang terdefinisi untuk menyelesaikan masalah
- Enam karakteristik algoritma: Input, Output, Definiteness, Finiteness, Effectiveness, Determinism
- Bahasa C adalah bahasa prosedural yang cocok untuk mempelajari dasar pemrograman
- Pemrograman prosedural fokus pada fungsi dan data, cocok untuk program sederhana
- Peta konsep menunjukkan alur dari algoritma, flowchart, variabel, percabangan, perulangan, hingga pengurutan dan pencarian

Kata Kunci: Algoritma, Pemrograman Prosedural, Bahasa C, Flowchart, Pseudocode

Bab 3

Pengenalan Algoritma dan Flowchart

Sub-CPMK yang Dicakup dalam Bab Ini:

- Sub-CPMK 1.1: Menjelaskan definisi dan fungsi flowchart
- Sub-CPMK 1.2: Membuat flowchart dan pseudocode untuk masalah sederhana

Materi Pokok: Definisi dan fungsi flowchart, simbol-simbol standar, langkah penyusunan flowchart, serta contoh penerapan pada masalah sederhana [11, 8].

3.1 Definisi dan Fungsi Flowchart

Flowchart adalah representasi visual dari algoritma menggunakan simbol-simbol standar yang dihubungkan dengan garis alur [11, 8]. Kata "flowchart" berasal dari "flow" (alur) dan "chart" (diagram), yang secara harfiah berarti diagram alur. Flowchart pertama kali diperkenalkan oleh Frank Gilbreth pada tahun 1921 untuk memvisualisasikan proses industri dan kemudian diadopsi dalam pemrograman komputer pada tahun 1960-an.

3.1.1 Fungsi Utama Flowchart

Flowchart memiliki beberapa fungsi penting dalam pengembangan perangkat lunak:

1. **Visualisasi Logika:** Mengubah algoritma abstrak menjadi diagram visual yang mudah dipahami
2. **Dokumentasi:** Membuat dokumentasi yang jelas tentang cara kerja program
3. **Komunikasi:** Memfasilitasi komunikasi antar tim programmer dan stakeholder
4. **Debugging:** Membantu mengidentifikasi kesalahan logika sebelum implementasi
5. **Analisis:** Memungkinkan analisis efisiensi dan kompleksitas algoritma

3.1.2 Manfaat Flowchart dalam Pembelajaran

Untuk mahasiswa pemula, flowchart memberikan beberapa keuntungan:

- **Pemahaman Konseptual:** Membantu memahami alur logika tanpa terbebani sintaks bahasa pemrograman
- **Problem Solving:** Melatih berpikir sistematis dalam memecahkan masalah
- **Struktur Berpikir:** Mengembangkan kemampuan berpikir terstruktur dan logis
- **Transisi ke Kode:** Menjadi jembatan antara konsep algoritma dan implementasi kode

3.1.3 Standar Flowchart

Flowchart mengikuti standar internasional yang ditetapkan oleh American National Standards Institute (ANSI) pada tahun 1970-an. Standar ini memastikan konsistensi dan universalitas dalam penggunaan simbol flowchart di seluruh dunia. Setiap simbol memiliki makna spesifik yang telah disepakati secara internasional, sehingga flowchart yang dibuat di Indonesia dapat dipahami oleh programmer di Amerika atau Eropa.

Penggunaan flowchart memudahkan komunikasi antar programmer dan pemangku kepentingan dalam memahami logika solusi sebelum implementasi teknis dilakukan.

3.2 Simbol Flowchart Standar

Berdasarkan standar ANSI dan referensi dari Lucidchart, simbol flowchart standar meliputi beberapa jenis dengan fungsi spesifik [8]. Setiap simbol dirancang untuk merepresentasikan jenis operasi tertentu dalam algoritma.

3.2.1 Simbol-Simbol Dasar

Simbol	**Nama**	**Fungsi dan Contoh**
Oval	Terminator	Mulai/Akhir program. Contoh: "Start", "End"
Persegi Panjang	Proses	Operasi, perhitungan, penugasan. Contoh: "x = x + 1"
Belah Ketupat	Keputusan	Percabangan (Ya/Tidak). Contoh: "x > 0?"
Jajar Genjang	Input/Output	Membaca input atau menampilkan output. Contoh: "Input nilai", "Print hasil"
Panah	Garis Alur	Menunjukkan urutan eksekusi program

Tabel 3.1: Simbol Flowchart Dasar

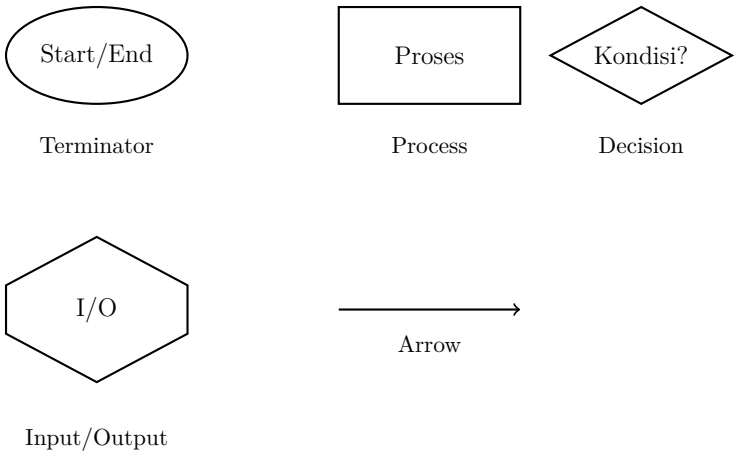
3.2.2 Simbol Tambahan

Simbol	**Nama**	**Fungsi dan Contoh**
Persegi Panjang Ganda	Preparasi	Persiapan atau inisialisasi. Contoh: "Inisialisasi variabel"
Hexagon	Loop	Awal dan akhir loop. Contoh: "For i = 1 to 10"
Lingkaran	Connector	Menghubungkan bagian flowchart yang sama halaman
Persegi Rumah	Off-page	Menghubungkan flowchart antar halaman
Garis Putus-putus	Comment	Komentar atau penjelasan tambahan
Dokumen	Data	Input/output dokumen. Contoh: "Baca file data.txt"

Tabel 3.2: Simbol Flowchart Tambahan

3.2.3 Contoh Visual Simbol Flowchart

Berikut adalah representasi visual dari simbol-simbol flowchart utama:



3.2.4 Aturan Penggunaan Simbol

- **Konsistensi:** Gunakan simbol yang sama untuk operasi yang sama di seluruh flowchart
- **Keterbacaan:** Pastikan teks di dalam simbol mudah dibaca dan singkat
- **Arah Alur:** Gunakan panah untuk menunjukkan arah alur yang jelas
- **Hierarki:** Simbol utama (terminator) selalu di bagian atas atau bawah
- **Koneksi:** Gunakan connector untuk flowchart yang kompleks

Pemahaman yang baik tentang simbol-simbol ini menjadi fondasi untuk membuat flowchart yang efektif dan sesuai standar industri.

3.3 Membuat Flowchart Sederhana

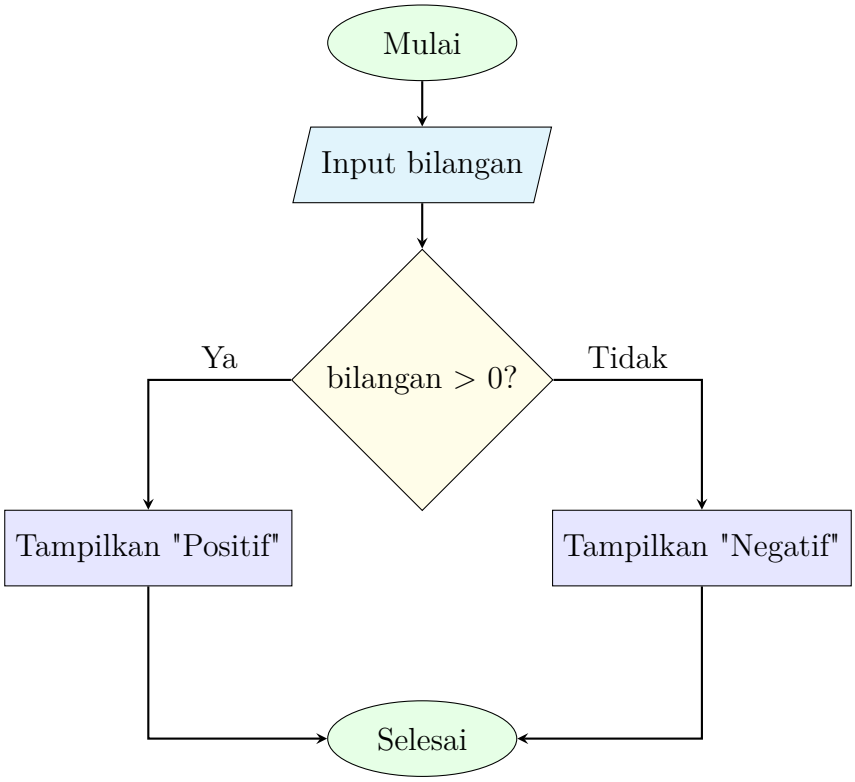
Untuk membuat flowchart, mulailah dengan simbol terminator bertuliskan "Mulai", kemudian tambahkan langkah-langkah algoritma secara berurutan menggunakan simbol proses atau input/output. Jika algoritma memiliki percabangan, gunakan simbol keputusan dengan dua cabang (Ya dan Tidak) yang menuju ke langkah yang sesuai. Akhiri flowchart dengan simbol terminator bertuliskan "Selesai". Pastikan setiap panah mengalir dengan jelas dan tidak ada langkah yang terputus.

3.3.1 Contoh 1: Menentukan Bilangan Positif atau Negatif

Algoritma:

1. Mulai
2. Input bilangan
3. Jika bilangan > 0 maka tampilkan "Positif"
4. Jika tidak maka tampilkan "Negatif"
5. Selesai

Flowchart:

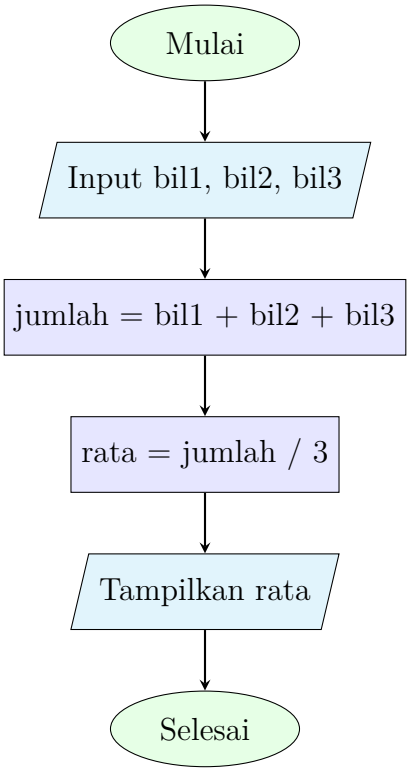


3.3.2 Contoh 2: Menghitung Rata-rata Tiga Bilangan

Algoritma:

1. Mulai
2. Input bilangan1, bilangan2, bilangan3
3. Hitung jumlah = bilangan1 + bilangan2 + bilangan3
4. Hitung rata-rata = jumlah / 3
5. Tampilkan rata-rata
6. Selesai

Flowchart:



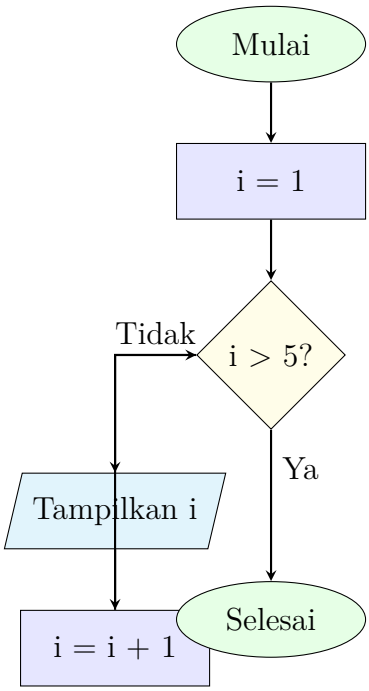
3.3.3 Contoh 3: Loop untuk Menampilkan Bilangan 1-5

Algoritma:

1. Mulai
2. Inisialisasi i = 1
3. Jika i > 5 maka ke langkah 7
4. Tampilkan i

5. i = i + 1
6. Kembali ke langkah 3
7. Selesai

Flowchart:



3.3.4 Tips Membuat Flowchart yang Baik

- **Mulai dari atas:** Alur flowchart sebaiknya mengalir dari atas ke bawah
- **Gunakan simbol yang tepat:** Pastikan setiap operasi menggunakan simbol yang sesuai
- **Hindari persimpangan:** Usahakan garis alur tidak saling memotong
- **Label yang jelas:** Gunakan teks yang singkat namun jelas
- **Uji logika:** Pastikan flowchart merepresentasikan algoritma dengan benar

3.4 Contoh Flowchart: Konversi Suhu

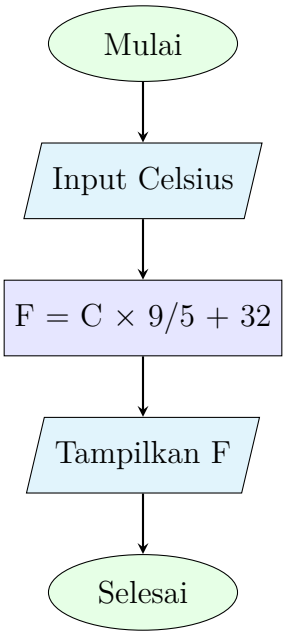
Berikut contoh flowchart lengkap untuk mengonversi suhu dari Celsius ke Fahrenheit dengan implementasi dalam bahasa C.

3.4.1 Algoritma Konversi Suhu

Algoritma:

1. Mulai
2. Input suhu dalam Celsius
3. Hitung Fahrenheit = Celsius $\times$ 9/5 + 32
4. Tampilkan hasil Fahrenheit
5. Selesai

3.4.2 Flowchart Lengkap



3.4.3 Implementasi dalam Bahasa C

```c
#include <stdio.h>

int main() {
    float celsius, fahrenheit;

    // Input suhu Celsius
    printf("Masukkan suhu dalam Celsius: ");
    scanf("%f", &celsius);

    // Konversi ke Fahrenheit
    fahrenheit = celsius * 9.0 / 5.0 + 32;

    // Tampilkan hasil
```

```
14      printf("%.2f Celsius = %.2f Fahrenheit\n", celsius,
            fahrenheit);
15
16      return 0;
17  }
```

Kode Program 3.1: Program Konversi Suhu Celsius ke Fahrenheit

3.4.4 Analisis Karakteristik Algoritma

- **Input:** Satu nilai suhu dalam Celsius
- **Output:** Satu nilai suhu dalam Fahrenheit
- **Definiteness:** Langkah-langkah jelas (input, hitung, output)
- **Finiteness:** Algoritma berakhir setelah 4 langkah
- **Effectiveness:** Operasi aritmatika dasar dapat dilakukan
- **Determinism:** Input 25°C selalu menghasilkan 77°F

3.4.5 Contoh Output Program

```
Masukkan suhu dalam Celsius: 25
25.00 Celsius = 77.00 Fahrenheit

Masukkan suhu dalam Celsius: 0
0.00 Celsius = 32.00 Fahrenheit

Masukkan suhu dalam Celsius: 100
100.00 Celsius = 212.00 Fahrenheit
```

Contoh ini menunjukkan bagaimana flowchart dapat membantu merancang algoritma sederhana sebelum implementasi dalam bahasa pemrograman.

Aktivitas Pembelajaran

1. Buat flowchart untuk menentukan bilangan ganjil atau genap.
2. Buat flowchart untuk menghitung luas persegi panjang dari input panjang dan lebar.
3. Buat flowchart untuk menampilkan bilangan 1 sampai 5.

Latihan dan Refleksi

1. Jelaskan fungsi masing-masing simbol flowchart: terminator, proses, keputusan, input/output!

2. Buat flowchart untuk algoritma konversi suhu Celsius ke Fahrenheit!

3. **Refleksi**: Apakah flowchart membantu Anda memahami alur algoritma? Jelaskan!

Asesmen (Evaluasi Kinerja)

Instrumen untuk Sub-CPMK 1.2: Buat flowchart untuk menentukan nilai maksimum dari dua bilangan yang diinput.

Checklist Pencapaian Kompetensi

Centang item berikut setelah Anda yakin telah menguasainya:

☐ Saya dapat menjelaskan definisi dan fungsi flowchart

☐ Saya dapat mengenali simbol flowchart standar

☐ Saya dapat membuat flowchart untuk masalah sederhana

Rangkuman

Bab ini membahas flowchart sebagai representasi visual algoritma, simbol standar (terminator, proses, keputusan, input/output), dan cara membuat flowchart untuk masalah sederhana.

Bab 4

Pseudocode dan Notasi Algoritma

Sub-CPMK yang Dicakup dalam Bab Ini:

- Sub-CPMK 1.2: Membuat pseudocode untuk masalah sederhana dan mengkonversinya ke bahasa C

Materi Pokok: Konsep dan notasi pseudocode, pemetaan pseudocode ke struktur kontrol C, serta contoh konversi sederhana [16, 3].

4.1 Definisi dan Notasi Pseudocode

Pseudocode adalah deskripsi algoritma dalam bahasa manusia yang menyerupai kode program namun tidak terikat pada sintaks bahasa pemrograman tertentu [16]. Kata "pseudocode" berasal dari "pseudo" (semu) dan "code" (kode), yang berarti kode semu atau tiruan kode program. Pseudocode pertama kali digunakan pada tahun 1950-an sebagai alat untuk mendokumentasikan algoritma sebelum implementasi dalam bahasa assembly.

4.1.1 Fungsi Pseudocode

Pseudocode memiliki beberapa fungsi penting dalam pengembangan perangkat lunak:

1. **Perancangan Logika:** Memungkinkan fokus pada algoritma tanpa detail sintaks
2. **Dokumentasi:** Membuat dokumentasi algoritma yang mudah dipahami
3. **Komunikasi:** Memfasilitasi komunikasi tim tanpa terikat bahasa pemrograman
4. **Validasi:** Memungkinkan pengecekan logika sebelum implementasi
5. **Pembelajaran:** Membantu pemula memahami konsep pemrograman

4.1.2 Standar Notasi Pseudocode

Berikut adalah standar notasi pseudocode yang umum digunakan:

Kategori	**Notasi Pseudocode**	**Contoh**
Struktur	MULAI, SELESAI	MULAI ... SELESAI
Input/Output	BACA, CETAK, TULIS	BACA nilai, CETAK "Hello"
Assignment	:= atau =	nilai := 10, x = y + 1
Percabangan	JIKA...MAKA...JIKA TIDAK	JIKA x > 0 MAKA CETAK "Positif"
Perulangan	UNTUK...DARI...SAMPAI	UNTUK i := 1 DARI 1 SAMPAI 10
Perulangan	SELAMA...LAKUKAN	SELAMA $x > 0$ LAKUKAN $x := x - 1$
Prosedur	PROSEDUR nama()	PROSEDUR hitungRata()
Fungsi	FUNGSI nama()	FUNGSI maks(a, b)
Komentar	// atau /* ... */	// Ini komentar

Tabel 4.1: Standar Notasi Pseudocode

4.1.3 Aturan Penulisan Pseudocode

- **Konsistensi:** Gunakan notasi yang sama di seluruh pseudocode
- **Indentasi:** Gunakan indentasi untuk menunjukkan blok kode
- **Bahasa Jelas:** Gunakan bahasa Indonesia atau Inggris yang jelas
- **Struktur Logis:** Ikuti alur logika yang sistematis
- **Detail Tepat:** Berikan detail cukup namun tidak terlalu teknis

4.1.4 Perbandingan Pseudocode dan Flowchart

Pseudocode	**Flowchart**
Teks-based, linear	Visual, grafis
Lebih detail implementasi	Lebih fokus pada alur logika
Mudah diketik	Membutuhkan alat gambar
Baik untuk algoritma kompleks	Baik untuk presentasi
Struktur hierarki jelas	Alur visual langsung

Tabel 4.2: Perbandingan Pseudocode dan Flowchart

Pseudocode memudahkan perancangan logika algoritma sebelum implementasi ke bahasa pemrograman seperti C. Dengan pseudocode, programmer dapat fokus pada alur logika tanpa khawatir tentang detail sintaks.

4.2 Contoh Pseudocode dan Konversi ke Bahasa C

Berikut adalah contoh-contoh pseudocode lengkap dengan konversinya ke bahasa C untuk memahami hubungan antara desain algoritma dan implementasi kode.

4.2.1 Contoh 1: Program Hello World

Pseudocode:

```
MULAI
    CETAK "Hello, World"
SELESAI
```

Konversi ke C:

```c
#include <stdio.h>

int main() {
    printf("Hello, World\n");
    return 0;
}
```

Kode Program 4.1: Konversi Hello World ke C

Analisis Konversi:

- MULAI → `int main() {`
- CETAK → `printf()`
- SELESAI → `return 0; }`

4.2.2 Contoh 2: Membaca Input dan Menampilkan Output

Pseudocode:

```
MULAI
    BACA nama
    CETAK "Halo, ", nama
SELESAI
```

Konversi ke C:

```c
#include <stdio.h>

int main() {
    char nama[50];

    printf("Masukkan nama: ");
    scanf("%s", nama);
    printf("Halo, %s\n", nama);
```

```
9
10      return 0;
11  }
```

Kode Program 4.2: Konversi Input/Output ke C

Analisis Konversi:

- BACA nama → `scanf("%s", nama)`
- CETAK → `printf()`
- Perlu deklarasi variabel: `char nama[50];`

4.2.3 Contoh 3: Percabangan If-Else

Pseudocode:

```
MULAI
    BACA nilai
    JIKA nilai > 60 MAKA
        CETAK "Lulus"
    JIKA TIDAK
        CETAK "Tidak Lulus"
SELESAI
```

Konversi ke C:

```
#include <stdio.h>

int main() {
    int nilai;

    printf("Masukkan nilai: ");
    scanf("%d", &nilai);

    if (nilai > 60) {
        printf("Lulus\n");
    } else {
        printf("Tidak Lulus\n");
    }

    return 0;
}
```

Kode Program 4.3: Konversi Percabangan ke C

4.2.4 Contoh 4: Perulangan For

Pseudocode:

```
MULAI
    UNTUK i := 1 DARI 1 SAMPAI 5 LAKUKAN
        CETAK i
SELESAI
```

Konversi ke C:

```c
#include <stdio.h>

int main() {
    int i;

    for (i = 1; i <= 5; i++) {
        printf("%d\n", i);
    }

    return 0;
}
```

Kode Program 4.4: Konversi Perulangan ke C

4.2.5 Contoh 5: Menghitung Rata-rata

Pseudocode:

```
MULAI
    BACA bil1, bil2, bil3
    jumlah := bil1 + bil2 + bil3
    rata := jumlah / 3
    CETAK "Rata-rata: ", rata
SELESAI
```

Konversi ke C:

```c
#include <stdio.h>

int main() {
    float bil1, bil2, bil3, jumlah, rata;

    printf("Masukkan tiga bilangan: ");
    scanf("%f %f %f", &bil1, &bil2, &bil3);

    jumlah = bil1 + bil2 + bil3;
    rata = jumlah / 3;

    printf("Rata-rata: %.2f\n", rata);

    return 0;
}
```

Kode Program 4.5: Konversi Perhitungan Rata-rata ke C

4.2.6 Tabel Pemetaan Konversi

Pseudocode	**Bahasa C**
MULAI	`int main() {`
SELESAI	`return 0; }`
BACA variabel	`scanf("%format", &variabel);`
CETAK teks	`printf("teks\n");`
variabel := nilai	`variabel = nilai;`
JIKA kondisi MAKA	`if (kondisi) {`
JIKA TIDAK	`} else {`
UNTUK i := 1 SAMPAI n	`for (i = 1; i <= n; i++) {`
SELAMA kondisi	`while (kondisi) {`
LAKUKAN	`// Isi loop`

Tabel 4.3: Pemetaan Konversi Pseudocode ke Bahasa C

Konversi pseudocode ke C dilakukan dengan memetakan setiap notasi pseudocode ke sintaks yang sesuai dalam bahasa C, ditambah dengan deklarasi variabel dan header yang diperlukan.

Aktivitas Pembelajaran

1. Tulis pseudocode untuk menghitung rata-rata tiga bilangan, lalu konversikan ke C.

2. Bandingkan flowchart dan pseudocode untuk masalah yang sama. Mana yang lebih mudah dibuat?

Latihan dan Refleksi

1. Jelaskan perbedaan flowchart dan pseudocode!

2. Tulis pseudocode untuk menentukan bilangan terbesar dari tiga input!

3. **Refleksi**: Bagian mana dari konversi pseudocode ke C yang paling menantang bagi Anda? Jelaskan alasannya.

Asesmen (Evaluasi Kinerja)

Instrumen untuk Sub-CPMK 1.2: Buat pseudocode dan implementasi C untuk meng-

hitung luas segitiga dari input alas dan tinggi, lalu jelaskan pemetaan setiap langkah pseudocode ke kode C.

Checklist Pencapaian Kompetensi

Centang item berikut setelah Anda yakin telah menguasainya:

☐ Saya dapat menulis pseudocode untuk masalah sederhana

☐ Saya dapat mengkonversi pseudocode ke kode C

Rangkuman

Pseudocode adalah deskripsi algoritma dalam bahasa manusia yang memudahkan perancangan sebelum implementasi ke C.

Bab 5

Variabel, Tipe Data, dan I/O dalam C

Sub-CPMK yang Dicakup dalam Bab Ini:

- Sub-CPMK 3.1 (dasar): Mendeklarasikan variabel, menggunakan tipe data, serta `printf` dan `scanf`

Materi Pokok: Konsep variabel, tipe data dasar, format I/O dengan `printf`/`scanf`, serta casting tipe data [3].

5.1 Variabel dan Deklarasi dalam C

Variabel adalah tempat penyimpanan data di memori yang memiliki nama dan tipe data [3]. Dalam pemrograman, variabel berfungsi seperti wadah yang dapat menampung nilai-nilai yang berubah selama eksekusi program. Setiap variabel dalam C memiliki alamat memori unik yang digunakan untuk menyimpan dan mengakses data.

5.1.1 Deklarasi Variabel

Dalam C, variabel harus dideklarasikan sebelum digunakan dengan sintaks umum:

```
tipe_data nama_variabel;
```

Contoh Deklarasi:

```
int umur;                    // Variabel integer untuk umur
float tinggi;                // Variabel float untuk tinggi
    badan
char grade;                  // Variabel karakter untuk nilai
double gpa;                  // Variabel double untuk IPK
```

5.1.2 Inisialisasi Variabel

Variabel dapat diinisialisasi saat deklarasi atau diberi nilai kemudian:

```c
// Inisialisasi saat deklarasi
int x = 10;
float pi = 3.14159;
char huruf = 'A';

// Inisialisasi terpisah
int y;
y = 20;  // Assignment setelah deklarasi
```

5.1.3 Aturan Penamaan Variabel

Nama variabel (identifier) harus mengikuti aturan berikut:

- **Karakter yang diizinkan:** Huruf (a-z, A-Z), angka (0-9), dan underscore (_)
- **Karakter pertama:** Harus huruf atau underscore, tidak boleh angka
- **Panjang maksimal:** Biasanya 31 karakter (standar C)
- **Case sensitive:** `nilai` berbeda dengan `Nilai`
- **Reserved words:** Tidak boleh menggunakan kata kunci C seperti `int`, `if`, `while`

Contoh nama variabel yang valid:

```c
int umurMahasiswa;
float nilai_rata_rata;
char grade;
int _counter;
```

Contoh nama variabel yang tidak valid:

```c
int 2umur;           // Tidak boleh diawali angka
float nilai-rata;    // Tidak boleh menggunakan strip (-)
char class;          // Tidak boleh reserved word
```

5.1.4 Scope Variabel

Scope menentukan di mana variabel dapat diakses dalam program:

Contoh Scope:

```c
#include <stdio.h>

int globalVar = 100;  // Variabel global

```

Jenis Scope	Keterangan
Local	Dideklarasikan di dalam fungsi, hanya dapat diakses di fungsi tersebut
Global	Dideklarasikan di luar semua fungsi, dapat diakses di seluruh program
Block	Dideklarasikan di dalam blok {}, hanya dapat diakses di blok tersebut

Tabel 5.1: Jenis Scope Variabel dalam C

```c
int main() {
    int localVar = 50;  // Variabel local

    {
        int blockVar = 25;  // Variabel block
        printf("Block: %d\n", blockVar);  // Bisa diakses
    }

    printf("Local: %d\n", localVar);    // Bisa diakses
    printf("Global: %d\n", globalVar);   // Bisa diakses
    // printf("Block: %d\n", blockVar);  // Error: tidak bisa
       diakses

    return 0;
}
```

5.2 Tipe Data Dasar

Tipe data dasar C menentukan jenis data yang dapat disimpan dalam variabel, ukuran memori yang digunakan, dan rentang nilai yang mungkin. Pemilihan tipe data yang tepat sangat penting untuk efisiensi memori dan akurasi perhitungan.

5.2.1 Tipe Data Integer

Tipe data integer digunakan untuk menyimpan bilangan bulat (tanpa desimal):

Tipe Data	Ukuran	Rentang Nilai	Contoh Penggunaan
`short int`	2 byte	-32,768 hingga 32,767	Usia, jumlah kecil
`unsigned short`	2 byte	0 hingga 65,535	ID positif
`int`	4 byte	-2,147,483,648 hingga 2,147,483,647	Counter, umum
`unsigned int`	4 byte	0 hingga 4,294,967,295	ID, populasi
`long int`	4/8 byte	Sama seperti int	Kompatibilitas
`long long`	8 byte	$\pm 9.2 \times 10^{18}$	Bilangan sangat besar

Tabel 5.2: Tipe Data Integer dalam C

5.2.2 Tipe Data Floating Point

Tipe data floating point digunakan untuk menyimpan bilangan desimal:

Tipe Data	Ukuran	Presisi	Contoh Penggunaan
`float`	4 byte	6-7 digit desimal	Suhu, persentase
`double`	8 byte	15-16 digit desimal	IPK, nilai akurat
`long double`	10/16 byte	19-21 digit desimal	Ilmiah, presisi tinggi

Tabel 5.3: Tipe Data Floating Point dalam C

5.2.3 Tipe Data Karakter

- **`char`:** Menyimpan satu karakter (1 byte)
- **Rentang:** -128 hingga 127 (signed) atau 0 hingga 255 (unsigned)
- **Penggunaan:** Huruf, angka, simbol, atau nilai ASCII

Contoh:

```
char grade = 'A';          // Karakter A
char newline = '\n';       // Karakter newline
char nilai_ascii = 65;     // Sama dengan 'A'
```

5.2.4 Modifier Tipe Data

Modifier dapat digunakan untuk mengubah sifat tipe data dasar:

Modifier	Pengaruh
`signed`	Default, dapat menyimpan nilai positif dan negatif
`unsigned`	Hanya nilai non-negatif, rentang positif dua kali lipat
`short`	Mengurangi ukuran memori (biasanya setengah dari int)
`long`	Meningkatkan ukuran memori (minimal sama dengan int)

Tabel 5.4: Modifier Tipe Data dalam C

5.2.5 Memilih Tipe Data yang Tepat

Guidelines pemilihan tipe data:

- **Usia, jumlah item:** `int` atau `unsigned int`
- **Uang, nilai akademik:** `double` (hindari float untuk akurasi)

- **Suhu, persentase:** `float` (cukup untuk presisi biasa)
- **Huruf, status:** `char`
- **Bilangan sangat besar:** `long long`
- **ID yang selalu positif:** `unsigned int`

Contoh implementasi:

```c
#include <stdio.h>

int main() {
    // Tipe data yang tepat untuk berbagai kebutuhan
    int umur = 20;                      // Umur mahasiswa
    double ipk = 3.75;                   // IPK (perlu presisi)
    float suhu = 36.5f;                 // Suhu tubuh
    char grade = 'A';                    // Nilai huruf
    unsigned int npm = 2024001234;       // Nomor pokok
       mahasiswa

    printf("Umur: %d tahun\n", umur);
    printf("IPK: %.2f\n", ipk);
    printf("Suhu: %.1f derajat C\n", suhu);
    printf("Grade: %c\n", grade);
    printf("NPM: %u\n", npm);

    return 0;
}
```

5.3 Input dan Output

Fungsi `printf` digunakan untuk menampilkan output ke layar; format specifier seperti `%d` (integer), `%f` (float), `%c` (char), `%s` (string). Fungsi `scanf` digunakan untuk membaca input; memerlukan alamat variabel dengan operator `&`. Contoh: `scanf("%d", &x);` membaca integer ke variabel `x`. Header `stdio.h` harus disertakan untuk kedua fungsi.

5.4 Type Casting

Type casting mengubah tipe data suatu nilai ke tipe lain. Cast eksplisit: `(tipe) ekspresi`, misalnya `(float) 5 / 2` menghasilkan 2.5. Cast implisit terjadi saat tipe berbeda dalam operasi; nilai tipe lebih kecil dinaikkan ke tipe lebih besar.

Perhatikan kehilangan presisi saat casting dari float ke int. Contoh: `int a = (int) 3.14;` menghasilkan `a = 3`.

Aktivitas Pembelajaran

1. Buat program yang membaca nama dan umur, lalu menampilkannya.

2. Buat program konversi suhu Celsius ke Fahrenheit.

Latihan dan Refleksi

1. Jelaskan perbedaan `int`, `float`, dan `double`!

2. Mengapa `scanf` memerlukan `&` sebelum nama variabel?

3. **Refleksi**: Bagaimana Anda memilih tipe data yang paling tepat untuk sebuah masalah?

Asesmen (Evaluasi Kinerja)

Instrumen untuk Sub-CPMK 3.1: Buat program C yang membaca nama, usia, dan tinggi badan, lalu menampilkan kembali dengan format rapi menggunakan `printf`. Jelaskan alasan pemilihan tipe data dan format specifier.

Checklist Pencapaian Kompetensi

Centang item berikut setelah Anda yakin telah menguasainya:

☐ Saya dapat mendeklarasikan variabel dengan tipe data yang tepat

☐ Saya dapat menggunakan `printf` dan `scanf`

☐ Saya memahami type casting

Rangkuman

Variabel menyimpan data; tipe data dasar C meliputi int, float, double, char. `printf` dan `scanf` digunakan untuk I/O.

Bab 6

Operator dalam C

Sub-CPMK yang Dicakup dalam Bab Ini:

- Sub-CPMK 3.1: Menggunakan operator aritmatika, perbandingan, logika, bitwise, dan penugasan secara tepat dalam menyelesaikan masalah sederhana.

Materi Pokok:

- Operator Aritmatika dan Increment/Decrement
- Operator Relasional dan Logika
- Operator Bitwise dan Penugasan
- Prioritas Operator (Precedence) dan Asosiativitas

Referensi: [3, 4, 2, 7]

6.1 Operator Aritmatika dan Increment/Decrement

Operator adalah simbol khusus yang memberitahu kompilator untuk melakukan operasi matematika atau logika tertentu. Bahasa C kaya akan operator internal.

6.1.1 Operator Aritmatika

Operator aritmatika digunakan untuk melakukan operasi matematika dasar.

Catatan Penting:

1. **Pembagian Integer**: Jika kedua operand adalah integer, hasilnya adalah integer (dibulatkan ke bawah/truncate). Contoh: `7 / 2` menghasilkan `3`, bukan `3.5`. Agar

Tabel 6.1: Operator Aritmatika dalam C

Operator	Nama	Contoh	Keterangan
`+`	Penjumlahan	`a + b`	Menjumlahkan dua operand
`-`	Pengurangan	`a - b`	Mengurangkan operand kedua dari pertama
`*`	Perkalian	`a * b`	Mengalikan dua operand
`/`	Pembagian	`a / b`	Membagi operand pertama dengan kedua
`%`	Modulus (Sisa Bagi)	`a % b`	Sisa hasil bagi integer

mendapatkan hasil desimal, salah satu operand harus bertipe *floating point* (misal: `7.0 / 2`).

2. **Modulus**: Operator `%` hanya bekerja pada tipe data integer. `7 % 2` bernilai `1`. Pada C modern, tanda hasil modulus mengikuti operand pertama (misal `-7 % 3` hasilnya `-1`).

6.1.2 Operator Increment dan Decrement

C memiliki operator unik untuk menambah atau mengurangi nilai variabel sebesar 1, yaitu `++` dan `-`. Operator ini bisa diletakkan sebelum variabel (*prefix*) atau sesudah variabel (*postfix*).

- **Prefix (`++a`)**: Nilai variabel diubah *terlebih dahulu*, lalu hasilnya digunakan dalam ekspresi.
- **Postfix (`a++`)**: Nilai variabel *saat ini* digunakan dulu dalam ekspresi, baru kemudian nilainya diubah.

```c
#include <stdio.h>

int main() {
    int a = 5, b = 5;
    int hasil_a, hasil_b;

    // Prefix increment
    hasil_a = ++a;
    // a bertambah jadi 6 dulu, lalu nilai 6 dimasukkan ke hasil_a
    printf("a: %d, hasil_a: %d\n", a, hasil_a); // Output: 6, 6

    // Postfix increment
    hasil_b = b++;
    // nilai b (5) dimasukkan ke hasil_b dulu, baru b bertambah jadi 6
    printf("b: %d, hasil_b: %d\n", b, hasil_b); // Output: 6, 5

    return 0;
```

```
18 }
```

Kode Program 6.1: Perbedaan Prefix dan Postfix

6.1.3 Operator Unary

Operator unary hanya memerlukan satu operand. Selain increment/decrement, ada:

- `+` (Unary Plus): Menandakan nilai positif (jarang ditulis eksplisit).
- `-` (Unary Minus): Mengnegasikan nilai (mengubah positif jadi negatif, dan sebaliknya).

Operator aritmatika C: `+`, `-`, `*`, `/`, `%` (modulo). Prioritas mengikuti aturan matematika. Operator perbandingan: `==`, `!=`, `<`, `>`, `<=`, `>=`; menghasilkan nilai 1 (true) atau 0 (false). Operator penugasan: `=` serta bentuk majemuk `+=`, `-=`, `*=`, `/=`. Hati-hati dengan `=` vs `==`; kesalahan umum menggunakan `=` pada kondisi.

6.2 Operator Relasional dan Logika

Operator relasional dan logika adalah fondasi dari pengambilan keputusan (percabangan) dan perulangan dalam pemrograman.

6.2.1 Operator Relasional (Perbandingan)

Operator ini membandingkan dua nilai dan menghasilkan nilai kebenaran. Di C (sebelum C99), nilai kebenaran direpresentasikan dengan integer: **0 untuk False** dan **1 (atau bukan nol) untuk True**.

Tabel 6.2: Operator Relasional

Operator	Deskripsi	Contoh (A=10, B=20)
`==`	Sama dengan	`(A == B)` → False (0)
`!=`	Tidak sama dengan	`(A != B)` → True (1)
`>`	Lebih besar dari	`(A > B)` → False (0)
`<`	Lebih kecil dari	`(A < B)` → True (1)
`>=`	Lebih besar atau sama dengan	`(A >= B)` → False (0)
`<=`	Lebih kecil atau sama dengan	`(A <= B)` → True (1)

Peringatan

Jangan tertukar antara operator penugasan `=` (assignment) dengan operator perbandingan `==` (equality). Menulis `if (x = 5)` adalah **valid** secara sintaks tapi seringkali berupa *bug logic*, karena akan melakukan assignment nilai 5 ke x, yang nilainya (5) dianggap True.

6.2.1.1 Membandingkan Floating Point

Hati-hati saat membandingkan tipe `float` atau `double` menggunakan `==` karena masalah presisi. Sebaiknya gunakan selisih absolut dengan nilai toleransi (epsilon).

```
// Kurang aman
if (f == 3.14) { ... }

// Lebih aman
if (fabs(f - 3.14) < 0.00001) { ... }
```

6.2.2 Operator Logika

Digunakan untuk menggabungkan beberapa ekspresi relasional.

Tabel 6.3: Tabel Kebenaran Operator Logika

A	B	A && B (AND)	A || B (OR)	!A (NOT)
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

6.2.2.1 Short-Circuit Evaluation

C menggunakan evaluasi *short-circuit* untuk efisiensi:

1. **AND (&&)**: Jika operand kiri **False** (0), operand kanan **tidak dievaluasi** karena hasilnya pasti False.
2. **OR (||)**: Jika operand kiri **True** (1), operand kanan **tidak dievaluasi** karena hasilnya pasti True.

Contoh keamanan menggunakan short-circuit:

```
// Mencegah pembagian dengan nol
if (x != 0 && (100 / x) > 10) {
    // Jika x == 0, bagian (100 / x) tidak akan dieksekusi
    // sehingga error division by zero terhindarkan.
}
```

Operator logika: `&&` (AND), `||` (OR), `!` (NOT). AND bernilai true hanya jika kedua operand true. OR bernilai true jika salah satu operand true. NOT membalik nilai kebenaran. Operator logika sering digunakan dalam kondisi percabangan dan perulangan. Contoh: `if (x > 0 && x < 10)` memeriksa apakah x berada di antara 0 dan 10. Short-circuit evaluation: jika operand pertama menentukan hasil, operand kedua tidak dievaluasi.

6.3 Operator Bitwise, Penugasan, dan Lainnya

Selain operator matematika dan logika standar, C menyediakan operator yang bekerja pada tingkat bit dan memori, yang menjadi salah satu kekuatan utama bahasa C.

6.3.1 Operator Bitwise

Operator bitwise bekerja pada level representasi biner dari integer. Operator ini tidak dapat digunakan pada tipe `float`, `double`, atau `long double`.

Tabel 6.4: Operator Bitwise

Op	Nama	Deskripsi
`&`	AND	Bit bernilai 1 jika kedua bit operand bernilai 1
`|`	OR	Bit bernilai 1 jika salah satu bit operand bernilai 1
`^`	XOR (Exclusive OR)	Bit bernilai 1 jika bit operand berbeda nilainya
`~`	NOT (Complement)	Membalik nilai bit (0 jadi 1, 1 jadi 0)
`«`	Left Shift	Menggeser bit ke kiri (mengalikan dengan 2^n)
`»`	Right Shift	Menggeser bit ke kanan (membagi dengan 2^n)

Contoh: Misalkan A = 60 (`0011 1100`) dan B = 13 (`0000 1101`).

- `A & B` = 12 (`0000 1100`)
- `A | B` = 61 (`0011 1101`)
- `A ^ B` = 49 (`0011 0001`)
- `A « 2` = 240 (`1111 0000`)

6.3.2 Operator Penugasan Majemuk (Compound Assignment)

C memungkinkan kita menggabungkan operasi aritmatika/bitwise dengan penugasan untuk penulisan kode yang lebih ringkas.

Tabel 6.5: Operator Penugasan Majemuk

Penulisan Singkat	Ekivalen Dengan
`a += b`	`a = a + b`
`a -= b`	`a = a - b`
`a *= b`	`a = a * b`
`a /= b`	`a = a / b`
`a %= b`	`a = a % b`
`a «= b`	`a = a « b`

6.3.3 Operator Lainnya

6.3.3.1 Operator Ternary (Conditional Operator)

Ini adalah satu-satunya operator C yang mengambil tiga operand. Bentuknya:

```
kondisi ? ekspresi_jika_true : ekspresi_jika_false
```

Contoh:

```c
int a = 10, b = 20;
int max;

// Menggunakan if-else biasa
if (a > b) max = a;
else max = b;

// Menggunakan ternary (lebih ringkas)
max = (a > b) ? a : b;
```

6.3.3.2 Operator `sizeof`

Operator unary yang mengembalikan ukuran dalam *bytes* dari tipe data atau variabel.

```c
printf("Ukuran int: %lu bytes\n", sizeof(int)); // Biasanya 4
```

6.3.3.3 Operator Koma (`,`)

Digunakan untuk memisahkan dua atau lebih ekspresi yang dievaluasi berurutan. Nilai ekspresi keseluruhan adalah nilai dari ekspresi paling kanan. Sering dijumpai dalam loop `for`.

```c
int x, y;
y = (x = 5, x + 10);
// x diisi 5, lalu hitung x+10 (15). y akan bernilai 15.
```

6.4 Prioritas Operator dan Studi Kasus

Dalam sebuah ekspresi kompleks yang memiliki banyak operator, urutan eksekusi ditentukan oleh **Prioritas Operator** (*Operator Precedence*) dan **Asosiativitas**.

6.4.1 Tabel Prioritas Operator

Tabel berikut menunjukkan urutan operator dari prioritas tertinggi (dievaluasi duluan) hingga terendah.

Tabel 6.6: Prioritas Operator (Disederhanakan)

Level	Operator	Deskripsi	Asos.
1	`() [] -> .`	Function call, Array subscript, Member	Kiri ke Kanan
2	`! ~ ++ - + - * & sizeof`	Unary operators	**Kanan ke Kiri**
3	`* / %`	Multiplicative	Kiri ke Kanan
4	`+ -`	Additive	Kiri ke Kanan
5	`« »`	Shift	Kiri ke Kanan
6	`< <= > >=`	Relational	Kiri ke Kanan
7	`== !=`	Equality	Kiri ke Kanan
8	`&`	Bitwise AND	Kiri ke Kanan
9	`^`	Bitwise XOR	Kiri ke Kanan
10	`|`	Bitwise OR	Kiri ke Kanan
11	`&&`	Logical AND	Kiri ke Kanan
12	`||`	Logical OR	Kiri ke Kanan
13	`? :`	Ternary	**Kanan ke Kiri**
14	`= += -=` dll	Assignment	**Kanan ke Kiri**
15	`,`	Comma	Kiri ke Kanan

6.4.2 Studi Kasus Evaluasi Ekspresi

Mari kita bedah cara C mengevaluasi ekspresi berikut:

```
hasil = 10 + 5 * 2 > 15 && 4 % 2 == 0;
```

Langkah Evaluasi:

1. **Aritmatika Tertinggi (*) dan (%):**
   - `5 * 2` menjadi `10`.
   - `4 % 2` menjadi `0`.
   - Ekspresi kini: `10 + 10 > 15 && 0 == 0`
2. **Aritmatika (+) dan (-):**
   - `10 + 10` menjadi `20`.
   - Ekspresi kini: `20 > 15 && 0 == 0`
3. **Relasional (>, <, >=, <=):**
   - `20 > 15` bernilai True (`1`).
   - Ekspresi kini: `1 && 0 == 0`
4. **Equality (==, !=):**
   - `0 == 0` bernilai True (`1`).
   - Sisa ekspresi: `1 && 1`

5. **Logika AND (&&):**
   - `1 && 1` bernilai True (`1`).
6. **Assignment (=):**
   - Nilai `1` disimpan ke variabel `hasil`.

Kesalahan Umum: Chaining Relational

Ekspresi matematika $5 < x < 10$ tidak bisa ditulis mentah-mentah di C sebagai `5 < x < 10`.

- C akan mengevaluasi `(5 < x)` terlebih dahulu, menghasilkan 0 atau 1.
- Kemudian hasil tersebut dibandingkan dengan 10: `(0 atau 1) < 10`, yang mana **selalu True**.
- **Solusi:** Gunakan operator logika: `5 < x && x < 10`.

Aktivitas Pembelajaran

1. **Eksperimen Increment**: Buat program kecil untuk membandingkan `a = ++b` dan `a = b++`. Tampilkan nilai variabels sebelum dan sesudah operasi.
2. **Bitwise Magic**: Gunakan operator bitwise `&` untuk memeriksa apakah sebuah bilangan ganjil atau genap (hint: cek bit terakhir).
3. **Short-Circuit Demo**: Buat kondisi `if` dengan dua ekspresi, di mana ekspresi kedua memiliki *side effect* (misal print teks atau increment). Amati apakah ekspresi kedua dijalankan jika ekspresi pertama sudah menentukan hasil (misal `FALSE && ...`).

Latihan dan Refleksi

1. Jelaskan perbedaan antara `x++` (postfix) dan `++x` (prefix) ketika digunakan dalam sebuah ekspresi!
2. Apa hasil dari ekspresi `10 % 3` dan `10 % -3`? Jelaskan perilaku operator modulus pada bilangan negatif di C.
3. Ubah kondisi berikut menjadi bentuk operator ternary:

```
1 if (nilai >= 60) status = 1;
2 else status = 0;
```

4. Diberikan `int a = 5; int b = 10;`. Berapa nilai `a` dan `b` setelah ekspresi: `int hasil = (a++ > 5) || (b++ > 10);`? Jelaskan mengapa `b` berubah atau tidak berubah.

5. **Refleksi**: Mengapa memahami prioritas operator (precedence) itu penting? Berikan contoh bug yang mungkin muncul jika kita mengabaikan prioritas operator.

Asesmen (Evaluasi Kinerja)

Instrumen untuk Sub-CPMK 3.1

Soal 1 (Aritmatika & Logika): Buat program C yang menerima input 3 sisi segitiga (a, b, c). Program harus menentukan:

1. Apakah ketiga sisi dapat membentuk segitiga yang valid? (Syarat: $a + b > c$ DAN $a + c > b$ DAN $b + c > a$)
2. Jika valid, tentukan jenisnya: Sama Sisi, Sama Kaki, atau Sembarang.

Gunakan operator logika dan relasional yang tepat.

Soal 2 (Bitwise): Buat program yang menerima input bilangan bulat, lalu menggunakan operator geser (« atau ») untuk mengalikan bilangan tersebut dengan 4 dan membaginya dengan 2. Bandingkan hasilnya dengan operator aritmatika biasa.

Soal 3 (Evaluasi Ekspresi): Evaluasi ekspresi berikut secara manual, lalu buat program C untuk memverifikasi jawaban Anda: `result = 10 + 5 * 2 > 15 && 4 % 2 == 0;` Jelaskan urutan pengerjaannya berdasarkan prioritas operator.

Checklist Pencapaian Kompetensi

Centang item berikut setelah Anda yakin telah menguasainya:

☐ Saya memahami perbedaan prefix dan postfix increment/decrement

☐ Saya dapat menggunakan operator aritmatika, relasional, dan logika dengan benar

☐ Saya memahami konsep *short-circuit evaluation*

☐ Saya dapat menggunakan operator bitwise dasar (&, |, ^, ~, «, »)

☐ Saya dapat menggunakan operator ternary sebagai alternatif if-else sederhana

☐ Saya memahami urutan evaluasi berdasarkan tabel prioritas operator

Rangkuman

Bab ini mengupas tuntas berbagai operator dalam bahasa C:

- **Aritmatika**: Melakukan perhitungan matematika (+, -, *, /, %). Perhatikan pembagian integer dan modulus.
- **Increment/Decrement**: Menambah/mengurang 1 nilai variabal. Prefix (`++i`) mengubah nilai sebelum digunakan, Postfix (`i++`) menggunakan nilai lama dulu baru mengubah.
- **Relasional & Logika**: Membandingkan nilai dan menggabungkan kondisi (&&, ||, !). Menggunakan *short-circuit evaluation* untuk efisiensi dan keamanan.
- **Bitwise**: Memanipulasi data pada level bit, sangat berguna untuk pemrograman sistem atau optimasi tingkat rendah.
- **Assignment Majemuk**: Menyingkat penulisan operasi update (`+=`, `*=`).
- **Ternary**: `kondisi ? nilai_true : nilai_false` untuk percabangan inline.
- **Precedence**: Aturan yang menentukan operator mana yang dieksekusi lebih dulu dalam ekspresi kompleks.

Bab 7

Struktur Percabangan (Selection)

Sub-CPMK yang Dicakup dalam Bab Ini:

- Sub-CPMK 2.1: Merancang algoritma dengan struktur seleksi yang tepat (if, if-else, nested if).
- Sub-CPMK 3.1: Mengimplementasikan struktur percabangan dalam bahasa C untuk menyelesaikan masalah pengambilan keputusan.

Materi Pokok:

- Konsep Percabangan: Single Selection, Two-Way Selection, Multi-Way Selection
- Struktur `if` dan `if-else`
- Struktur `if-else if` (Ladder)
- Percabangan Bersarang (*Nested If*)
- Studi Kasus dan Kesalahan Umum (*Common Pitfalls*)

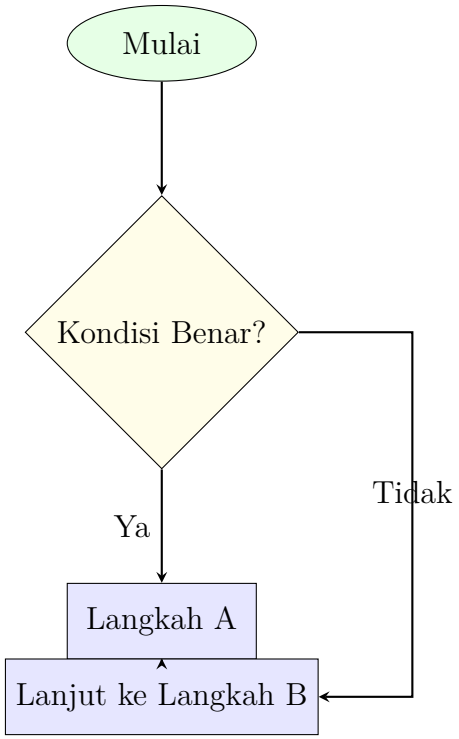
Referensi: [3, 4, 7]

7.1 Konsep Percabangan dan Struktur Dasar

Percabangan (branching) atau seleksi adalah salah satu elemen dasar algoritma yang memungkinkan program untuk melakukan tindakan yang berbeda berdasarkan kondisi tertentu. Tanpa percabangan, program hanya akan berjalan lurus (*sequential*) dari baris pertama hingga terakhir.

7.1.1 Alur Logika Seleksi

Dalam flowchart, keputusan digambarkan dengan simbol *Diamond* (Belah Ketupat). Simbol ini memiliki satu input aliran dan minimal dua output aliran (biasanya **True/Yes** dan **False/No**) [11].



Gambar 7.1: Flowchart Logika Single Selection (if)

7.1.2 Statement `if` (Single Selection)

Struktur `if` digunakan untuk menjalankan blok kode hanya jika kondisi bernilai **True** (non-zero). Jika kondisi **False** (0), blok kode dilewati.

Sintaks:

```
if (kondisi) {
    // Pernyataan yang dijalankan jika kondisi True
    pernyataan;
}
```

Contoh: Program menghitung nilai mutlak.

```
int angka = -5;
if (angka < 0) {
    angka = -angka; // Mengubah negatif menjadi positif
}
printf("Nilai mutlak: %d", angka); // Output: 5
```

7.1.3 Statement `if-else` (Two-Way Selection)

Struktur ini memberikan alternatif. Jika kondisi True, jalankan Blok A. Jika False, jalankan Blok B. Salah satu blok **pasti** dijalankan.

Sintaks:

```
if (kondisi) {
    // Blok A (Jika True)
} else {
    // Blok B (Jika False)
}
```

Contoh: Menentukan kelulusan.

```
if (nilai >= 60) {
    printf("Selamat, Anda Lulus!\n");
} else {
    printf("Maaf, Anda Harus Mengulang.\n");
}
```

7.1.4 Blok Kode dan Scope

Dalam C, blok kode ditandai dengan kurung kurawal `{ }`. Meskipun C mengizinkan penghilangan kurung kurawal jika blok hanya terdiri dari satu baris pernyataan, sangat disarankan untuk **selalu menggunakan kurung kurawal**.

```
// TIDAK DISARANKAN (Rawan Error)
if (x > 0)
    printf("Positif");
    x++; // Baris ini SELALU dijalankan, tidak terpengaruh if!

// DISARANKAN (Aman)
if (x > 0) {
    printf("Positif");
    x++; // Baris ini hanya jalan jika x > 0
}
```

Percabangan memungkinkan program mengambil keputusan berdasarkan kondisi. Sintaks `if`: `if (kondisi) { pernyataan; }`. Jika kondisi benar (non-nol), blok pernyataan dieksekusi. Sintaks `if-else`: `if (kondisi) { ... } else { ... }`; jika kondisi benar blok if dieksekusi, jika salah blok else dieksekusi. Blok dapat berisi satu atau lebih pernyataan; untuk satu pernyataan kurung kurawal bisa dihilangkan tapi tidak disarankan untuk kejelasan.

7.2 Percabangan Bertingkat dan Bersarang

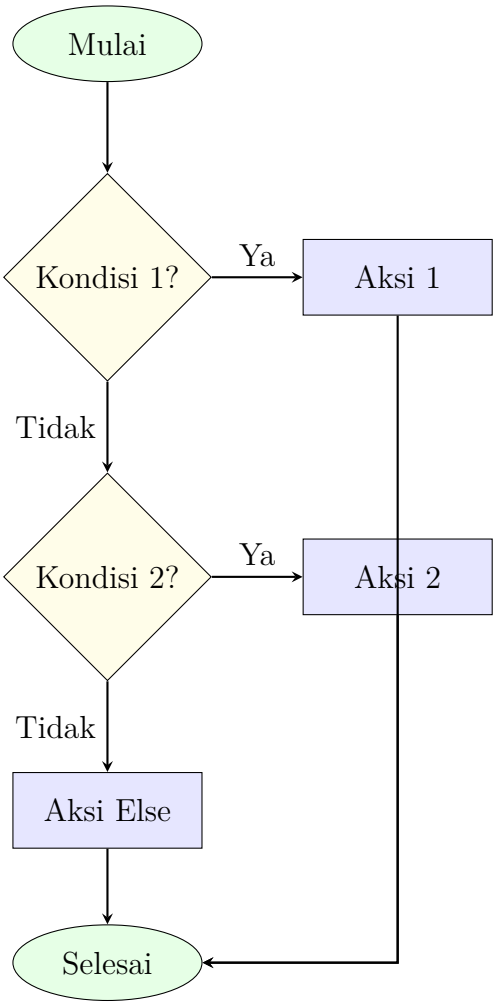
Seringkali keputusan yang diambil tidak cukup hanya ya/tidak, atau memiliki kondisi prasyarat. Untuk itu kita menggunakan percabangan bertingkat atau bersarang.

7.2.1 Struktur `if-else if` (Multi-Way Selection)

Digunakan ketika ada lebih dari dua kemungkinan kondisi yang harus diuji secara berurutan. Ini sering disebut sebagai T̈he if-else-if Ladder.̈

Prinsip Kerja:

1. Kondisi dievaluasi dari atas ke bawah.
2. Begitu ditemukan kondisi yang **True**, blok kodenya dieksekusi, dan sisa struktur dilewati (diabaikan).
3. Bagian `else` terakhir bersifat opsional, berfungsi sebagai b̈lock catch-allj̈ika tidak ada kondisi yang terpenuhi.



Gambar 7.2: Flowchart Bertingkat (Ladder If)

Contoh: Konversi Nilai Angka ke Huruf.

```
if (nilai >= 85) {
    printf("Grade A");
} else if (nilai >= 70) {
    printf("Grade B");
```

```
} else if (nilai >= 55) {
    printf("Grade C");
} else if (nilai >= 40) {
    printf("Grade D");
} else {
    printf("Grade E");
}
```

Pentingnya Urutan

Pada struktur tangga (*ladder*), urutan kondisi sangat krusial. Jika kita membalik urutannya: `if (nilai >= 40) ...` `else if (nilai >= 85) ...` Maka input nilai 90 akan masuk ke blok pertama (`>= 40` bernilai True), dan mencetak Grade D, yang mana salah. **Pastikan urutan kondisi logis (misal dari terbesar ke terkecil).**

7.2.2 Percabangan Bersarang (*Nested if*)

Kita bisa menempatkan struktur `if` di dalam blok `if` lainnya. Ini digunakan untuk logika berlapis atau prasyarat.

Sintaks:

```
if (kondisi1) {
    // Dieksekusi jika kondisi1 True
    if (kondisi2) {
        // Dieksekusi jika kondisi1 True DAN kondisi2 True
    } else {
        // Dieksekusi jika kondisi1 True DAN kondisi2 False
    }
} else {
    // Dieksekusi jika kondisi1 False
}
```

Contoh: Logika Login Sederhana.

```
if (username_valid) {
    if (password_valid) {
        printf("Login Berhasil!");
        if (is_admin) {
            printf("Selamat Datang Admin.");
        }
    } else {
        printf("Password Salah!");
    }
} else {
    printf("Username tidak ditemukan.");
}
```

7.2.2.1 Tips Mengelola Nesting

Nesting yang terlalu dalam (*Deep Nesting*) membuat kode sulit dibaca dan dipahami (*Spaghetti Code*).

- Gunakan operator logika (`&&`) untuk menggabungkan kondisi jika memungkinkan.
- Gunakan Ëarly Returnätau G̈uard Clause(akan dibahas di bab Fungsi).

Untuk kondisi bertingkat gunakan `if-else if-else`: `if (k1) ... else if (k2) ... else ...`. Kondisi dievaluasi berurutan; blok pertama yang memenuhi kondisi akan dieksekusi. Percabangan bersarang: if di dalam if. Indentasi yang baik penting untuk keterbacaan. Contoh: menentukan grade nilai (A, B, C, D, E) berdasarkan rentang skor menggunakan if-else if.

7.3 Kesalahan Umum dan Praktik Terbaik

Dalam menggunakan struktur percabangan, programmer pemula seringkali terjebak dalam beberapa kesalahan umum yang dapat menyebabkan *logic error* yang sulit dilacak.

7.3.1 Masalah *Dangling Else*

Masalah ini muncul ketika ada ‘if‘ bersarang namun jumlah ‘else‘ lebih sedikit daripada ‘if‘, dan tidak menggunakan kurung kurawal. Compiler C akan memasangkan ‘else‘ dengan ‘if‘ **terdekat sebelumnya**.

Kode Membingungkan (Tanpa Kurung Kurawal):

```c
if (a > 0)
    if (b > 0)
        printf("A dan B positif");
else
    printf("???"); // Else ini milik siapa?
```

Secara indentasi, seolah-olah ‘else‘ milik ‘if (a $>$ 0)‘. Namun bagi Compiler C, ‘else‘ tersebut milik ‘if (b $>$ 0)‘. Jika ‘a = -5‘, tidak ada output yang muncul (padahal mungkin kita mengharapkan blok else jalan).

Solusi: Selalu gunakan kurung kurawal!

```c
if (a > 0) {
    if (b > 0) {
        printf("A dan B positif");
    }
} else {
    printf("A negatif atau nol");
}
```

7.3.2 Assignment di dalam Kondisi

Kesalahan menggunakan '=' (penugasan) alih-alih '==' (perbandingan).

```
// SALAH - Bug Fatal
if (skor = 100) { // Nilai 100 di-assign ke skor. 100 != 0, maka dianggap True.
    printf("Sempurna!"); // Selalu tercetak, berapapun skor awalnya.
}

// BENAR
if (skor == 100) {
    printf("Sempurna!");
}
```

Tip

Beberapa programmer menggunakan gaya Yoda Condition: `if (100 == skor)` untuk mencegah error ini. Jika tidak sengaja menulis `if (100 = skor)`, compiler akan error karena literal angka tidak bisa di-assign.

7.3.3 Titik Koma (Semicolon) yang Salah Tempat

Menaruh titik koma tepat setelah 'if' akan mengakhiri statement 'if' tersebut tanpa menjalankan blok apa-apa.

```
// SALAH
if (nilai > 60); // Titik koma ini mengakhiri if!
{
    printf("Lulus"); // Blok ini jadi tidak terikat if, SELALU dijalankan.
}
```

7.3.4 Membandingkan Floating Point

Seperti dibahas di Bab 6, hindari menggunakan '==' untuk 'float' atau 'double'.

```
float x = 1.0 / 3.0;
if (x * 3.0 == 1.0) // Mungkin False karena presisi
```

Gunakan toleransi epsilon `fabs(a - b) < 0.00001`.

7.4 Studi Kasus Percabangan Kompleks

Bagian ini menyajikan studi kasus implementasi logika bisnis yang lebih kompleks menggunakan gabungan operator logika dan struktur percabangan.

7.4.1 Kasus 1: Menentukan Tahun Kabisat

Dulu kita diajarkan T̈ahun kabisat adalah tahun yang habis dibagi 4.̈ Namun aturan Gregorian Calendar sebenarnya:

1. Jika tahun habis dibagi 400, maka **Kabisat**.
2. Jika tidak habis dibagi 400 tetapi habis dibagi 100, maka **Bukan Kabisat**.
3. Jika tidak habis dibagi 100 tetapi habis dibagi 4, maka **Kabisat**.
4. Sisanya **Bukan Kabisat**.

Implementasi Nested If:

```
if (tahun % 400 == 0) {
    printf("Kabisat");
} else if (tahun % 100 == 0) {
    printf("Bukan Kabisat");
} else if (tahun % 4 == 0) {
    printf("Kabisat");
} else {
    printf("Bukan Kabisat");
}
```

Implementasi Single Logic (Efisiensi):

```
// Logika: (Habis 400) ATAU (Habis 4 DAN Tidak Habis 100)
if ((tahun % 400 == 0) || ((tahun % 4 == 0) && (tahun % 100 !=
   0))) {
    printf("Kabisat");
} else {
    printf("Bukan Kabisat");
}
```

7.4.2 Kasus 2: Validasi Segitiga

Diberikan tiga panjang sisi a, b, c. Tentukan apakah bisa membentuk segitiga, dan jika ya, jenisnya. Syarat Segitiga Valid (Triangle Inequality): $a + b > c$ DAN $a + c > b$ DAN $b + c > a$.

```
if (a + b > c && a + c > b && b + c > a) {
    // Segitiga Valid
    if (a == b && b == c) {
        printf("Segitiga Sama Sisi");
    } else if (a == b || a == c || b == c) {
        printf("Segitiga Sama Kaki");
    } else {
        // Cek Siku-Siku (Pythagoras)
        // Anggap c sisi terpanjang (perlu sorting idealnya)
        if (a*a + b*b == c*c || a*a + c*c == b*b || b*b + c*c
           == a*a)
```

```
            printf("Segitiga Siku-Siku");
        else
            printf("Segitiga Sembarang");
    }
} else {
    printf("Bukan Segitiga (Sisi tidak valid)");
}
```

7.4.3 Kasus 3: Menu Program Sederhana

Sebelum mengenal ‘switch-case‘, kita bisa membuat menu dengan ‘if-else if‘.

```
printf("Menu:\n1. Nasi Goreng\n2. Mie Goreng\n3. Jus Jeruk\n");
printf("Plihan Anda: ");
scanf("%d", &pilihan);

if (pilihan == 1) {
    harga = 15000;
    printf("Anda memesan Nasi Goreng. Harga: %d", harga);
} else if (pilihan == 2) {
    harga = 12000;
    printf("Anda memesan Mie Goreng. Harga: %d", harga);
} else if (pilihan == 3) {
    harga = 5000;
    printf("Anda memesan Jus Jeruk. Harga: %d", harga);
} else {
    printf("Menu tidak tersedia!");
}
```

Aktivitas Pembelajaran

1. **Analisis Logika**: Diberikan flowchart "Menentukan Bilangan Terbesar dari 3 Angka", tuliskan pseudocode dan implementasinya dalam C.

2. **Debugging**: Cari kesalahan pada kode berikut dan perbaiki:

```
if (nilai >= 70);
    printf("Lulus\n");
else
    printf("Tidak Lulus\n");
```

3. **Refactoring**: Ubah serangkaian `if` tunggal menjadi struktur `if-else if` yang lebih efisien untuk menentukan kategori usia (Anak, Remaja, Dewasa, Lansia).

Latihan dan Refleksi

1. Jelaskan perbedaan mendasar antara *Two-Way Selection* (`if-else`) dan *Multi-Way Selection* (`if-else if`)!

2. Apa itu *Dangling Else*? Bagaimana cara menghindarinya?

3. Buatlah tabel kebenaran untuk logika tahun kabisat (Kabisat jika: habis dibagi 400, ATAU (habis dibagi 4 TAPI tidak habis dibagi 100)).

4. **Refleksi**: Mengapa indentasi (penulisan kode yang menjorok) sangat penting dalam *Nested If*, meskipun compiler C tidak mempedulikannya?

Asesmen (Evaluasi Kinerja)

Instrumen untuk Sub-CPMK 2.1, 3.1

Soal 1 (Nested If - Kategori Nilai): Buat program yang membaca Nilai Angka (0-100) dan Kehadiran (0-100%). Mahasiswa dinyatakan LULUS jika Nilai Angka ≥ 60 DAN Kehadiran $\geq 80\%$.

- Jika Lulus, tentukan Grade: A (≥ 85), B (≥ 70), C (sisanya).
- Jika Tidak Lulus, berikan alasannya: "Nilai Kurang", "Kehadiran Kurang", atau "Keduanya Kurang".

Soal 2 (Validasi Segitiga): Baca 3 buah bilangan bulat positif. Tentukan apakah ketiga bilangan tersebut bisa membentuk segitiga. Jika ya, tentukan apakah segitiga Siku-siku (gunakan Pythagoras $a^2 + b^2 = c^2$), Sama Kaki, atau Sama Sisi.

Soal 3 (Analisis Kompleksitas): Diberikan 3 kondisi independen. Manakah yang lebih efisien: menggunakan 3 buah `if` terpisah atau menggunakan `if-else if`? Jelaskan alasannya.

Checklist Pencapaian Kompetensi

Centang item berikut setelah Anda yakin telah menguasainya:

☐ Saya memahami konsep blok kode (`{}`)

☐ Saya dapat menggunakan struktur `if` dan `if-else` dengan benar

☐ Saya dapat merancang logika bertingkat menggunakan `if-else if`

☐ Saya mampu mengelola kompleksitas pada *nested if*

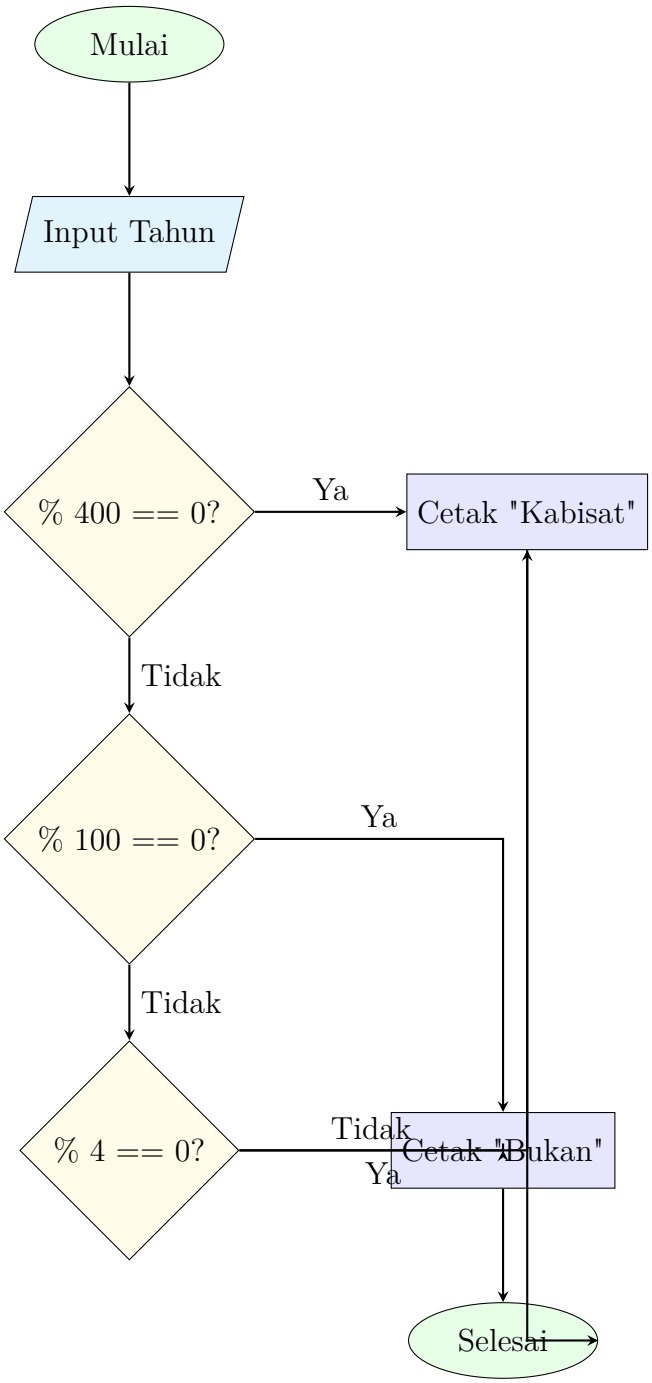
☐ Saya mengetahui bahaya *dangling else* dan assignment dalam kondisi

Rangkuman

Bab ini membahas struktur kontrol seleksi yang memungkinkan program berpikir dan

mengambil keputusan:

- `if`: Seleksi tunggal, hanya dijalankan jika kondisi True.
- `if-else`: Seleksi dua arah, memilih satu dari dua blok kode yang harus dijalankan.
- `if-else if`: Seleksi banyak arah (bertingkat), memeriksa kondisi secara berurutan sampai ada yang memenuhi.
- `Nested if`: Struktur percabangan di dalam percabangan lain untuk logika yang lebih spesifik/kompleks.
- **Best Practice**: Selalu gunakan kurung kurawal `{}` untuk setiap blok `if` atau `else` guna mencegah ambiguitas dan bug logis.



Gambar 7.3: Flowchart Logika Tahun Kabisat

Bab 8

Struktur Percabangan (switch-case)

Sub-CPMK yang Dicakup dalam Bab Ini:

- Sub-CPMK 3.1: Mengimplementasikan struktur seleksi switch-case dalam C

Materi Pokok: Sintaks `switch-case`, peran `break` dan `default`; kapan memilih switch daripada if-else [3, 4].

8.1 Sintaks switch-case

Struktur `switch-case` digunakan untuk percabangan berdasarkan nilai ekspresi yang dibandingkan dengan konstanta. Sintaks: `switch (ekspresi) { case nilai1: ... break; case nilai2: ... break; default: ... }`. Ekspresi harus bertipe integer atau char. Setiap `case` diikuti nilai konstanta dan titik dua. `break` mencegah fall-through ke case berikutnya. `default` menangani nilai yang tidak cocok dengan case mana pun.

8.2 Perbandingan switch dan if-else

Switch cocok ketika membandingkan satu variabel dengan banyak nilai konstanta diskret; kode lebih ringkas. If-else lebih fleksibel untuk kondisi kompleks, range nilai, atau ekspresi logika. Switch tidak bisa membandingkan string atau float secara langsung. Fall-through di switch (tanpa break) kadang disengaja untuk beberapa case yang menjalankan aksi sama. Pilih switch untuk menu pilihan angka/huruf; if-else untuk kondisi umum.

Aktivitas Pembelajaran

1. Buat program kalkulator dengan menu (1.Tambah 2.Kurang 3.Kali 4.Bagi) menggunakan switch.

2. Buat program konversi angka ke nama hari (1=Senin, 2=Selasa, ...).

Latihan dan Refleksi

1. Kapan sebaiknya menggunakan switch daripada if-else?

2. Apa fungsi break dalam switch? Apa yang terjadi jika dihilangkan?

3. **Refleksi**: Kapan Anda memilih menu dengan switch-case dan kapan dengan if-else? Jelaskan pertimbangan Anda.

Asesmen (Evaluasi Kinerja)

Instrumen untuk Sub-CPMK 3.1: Buat program C dengan menu (1–4) untuk operasi geometri: 1) luas persegi, 2) luas lingkaran, 3) luas segitiga, 4) keluar. Gunakan switch-case. Jelaskan mengapa switch tepat untuk kasus ini.

Checklist Pencapaian Kompetensi

Centang item berikut setelah Anda yakin telah menguasainya:

☐ Saya dapat menggunakan switch-case

☐ Saya mengetahui kapan menggunakan switch vs if-else

Rangkuman

Switch-case untuk percabangan berdasarkan nilai diskret. Gunakan break untuk mencegah fall-through; default untuk nilai tidak cocok.

Bab 9

Struktur Perulangan (for, while, do-while)

Sub-CPMK yang Dicakup dalam Bab Ini:

- Sub-CPMK 2.1, 3.2: Merancang dan mengimplementasikan struktur perulangan dalam C

Materi Pokok: Loop `for`, `while`, `do-while`; inisialisasi, kondisi, dan increment; kapan menggunakan masing-masing [3, 4].

9.1 Perulangan for dan while

Perulangan `for`: `for (inisialisasi; kondisi; update) { ... }`. Inisialisasi dijalankan sekali di awal; kondisi dicek setiap iterasi; update dijalankan setelah tiap iterasi. Perulangan `while`: `while (kondisi) { ... }`; tubuh loop dieksekusi selama kondisi benar. For cocok ketika jumlah iterasi diketahui; while cocok ketika kondisi berhenti tidak pasti. Pastikan ada mekanisme agar loop dapat berhenti untuk menghindari infinite loop.

9.2 Perulangan do-while

Perulangan `do-while`: `do { ... } while (kondisi);`. Berbeda dengan while, tubuh loop di `do-while` dieksekusi minimal sekali karena kondisi dicek di akhir. Berguna untuk input validasi: minta input berulang sampai valid. Sintaks: perhatikan titik koma setelah `while (kondisi)`. Bandingkan: while bisa tidak pernah dieksekusi; do-while selalu minimal satu kali.

Aktivitas Pembelajaran

1. Buat program menampilkan bilangan 1 sampai 10 dengan for.
2. Buat program menghitung jumlah digit bilangan dengan while.

Latihan dan Refleksi

1. Jelaskan perbedaan for, while, dan do-while!
2. Buat program menampilkan tabel perkalian!
3. **Refleksi**: Untuk tugas tampilkan bilangan 1 sampai N, mengapa for sering lebih nyaman daripada while? Kapan Anda justru memilih while?

Asesmen (Evaluasi Kinerja)

Instrumen untuk Sub-CPMK 2.1, 3.2: Buat program C yang membaca bilangan bulat positif N, lalu menampilkan jumlah bilangan dari 1 sampai N (1+2+...+N) menggunakan loop. Implementasikan dengan `for` dan bandingkan dengan versi `while`.

Checklist Pencapaian Kompetensi

Centang item berikut setelah Anda yakin telah menguasainya:

☐ Saya dapat menggunakan for, while, do-while

☐ Saya memahami kapan menggunakan masing-masing

Rangkuman

For untuk iterasi dengan jumlah pasti; while untuk kondisi berhenti; do-while minimal satu eksekusi.

Bab 10

Perulangan Bersarang

Sub-CPMK yang Dicakup dalam Bab Ini:

- Sub-CPMK 3.2: Mengimplementasikan perulangan bersarang

Materi Pokok: Perulangan bersarang (nested loop); alur eksekusi; penerapan untuk pola, tabel, dan matriks [3, 4].

10.1 Konsep Perulangan Bersarang

Perulangan bersarang (nested loop) adalah perulangan di dalam perulangan. Loop luar mengontrol baris; loop dalam mengontrol kolom. Setiap iterasi loop luar, loop dalam dijalankan lengkap. Contoh: menampilkan pola persegi atau segitiga dengan karakter. Perhatikan indeks dan kondisi masing-masing loop agar tidak konflik. Kompleksitas waktu bertambah: dua loop bersarang masing-masing n iterasi menghasilkan n^2 operasi.

10.2 Penerapan Perulangan Bersarang

Aplikasi: tabel perkalian (baris i, kolom j, tampilkan i*j), pola bintang (segitiga, persegi), operasi matriks (perkalian, transpose), array dua dimensi. Contoh pola segitiga: loop luar i dari 0 ke n, loop dalam j dari 0 ke i, cetak bintang. Untuk array 2D, loop bersarang untuk mengakses setiap elemen. Pahami urutan eksekusi: loop dalam selesai penuh sebelum loop luar increment.

Aktivitas Pembelajaran

1. Buat program menampilkan tabel perkalian 10x10.

2. Buat program menampilkan pola segitiga bintang.

Latihan dan Refleksi

1. Jelaskan alur eksekusi perulangan bersarang!

2. Buat program menampilkan pola piramida angka!

3. **Refleksi**: Saat membuat pola segitiga atau tabel, kesalahan apa yang paling sering Anda buat (indeks, kondisi)? Bagaimana Anda mendebugnya?

Asesmen (Evaluasi Kinerja)

Instrumen untuk Sub-CPMK 3.2: Buat program C yang membaca tinggi N, lalu menampilkan pola segitiga siku-siku terbalik dari bintang (tinggi N baris). Gunakan nested loop. Tuliskan pseudocode sebelum implementasi.

Checklist Pencapaian Kompetensi

Centang item berikut setelah Anda yakin telah menguasainya:

☐ Saya dapat membuat perulangan bersarang

☐ Saya dapat menerapkannya untuk pola dan tabel

Rangkuman

Perulangan bersarang: loop di dalam loop. Digunakan untuk pola, tabel, dan operasi matriks.

Bab 11

Fungsi dan Prosedur

Sub-CPMK yang Dicakup dalam Bab Ini:

- Sub-CPMK 4.1: Membuat fungsi dengan parameter dan nilai return dalam C

Materi Pokok: Definisi dan pemanggilan fungsi; parameter (pass by value) dan return value; scope variabel (lokal vs global) [3, 7].

11.1 Definisi Fungsi

Fungsi adalah blok kode yang dapat dipanggil berulang kali dengan nama. Fungsi mendukung modularisasi: memecah program menjadi bagian-bagian kecil yang dapat dipelihara. Sintaks definisi: `tipe_return nama(parameter) { ... return nilai; }`. Fungsi dengan tipe `void` tidak mengembalikan nilai. Parameter adalah input fungsi; bisa nol atau lebih, dipisah koma. Return mengembalikan nilai dan mengakhiri eksekusi fungsi.

11.2 Parameter dan Return Value

Parameter formal dideklarasikan dalam definisi fungsi; parameter aktual (argumen) diberikan saat pemanggilan. Pass by value: nilai argument disalin ke parameter; perubahan di dalam fungsi tidak mempengaruhi variabel asal. Untuk mengembalikan lebih dari satu nilai atau mengubah variabel asal, gunakan pointer (materi lanjut). Return value harus bertipe sesuai deklarasi fungsi. Contoh: `int tambah(int a, int b) { return a + b; }`.

11.3 Scope Variabel

Scope variabel menentukan di mana variabel dapat diakses. Variabel lokal dideklarasikan di dalam fungsi; hanya bisa diakses dalam fungsi itu. Variabel global dideklarasikan di luar semua fungsi; dapat diakses di mana saja. Hindari penggunaan global berlebihan; preferensi variabel lokal untuk modularitas. Lifetime: variabel lokal ada selama eksekusi fungsi; variabel global ada selama program berjalan. Variabel dengan nama sama: lokal menutupi (shadow) global dalam scope-nya.

Aktivitas Pembelajaran

1. Buat fungsi untuk menghitung faktorial, lalu panggil dari main.
2. Buat fungsi untuk mengecek bilangan prima.

Latihan dan Refleksi

1. Jelaskan perbedaan parameter dan return value!
2. Apa perbedaan variabel lokal dan global?
3. **Refleksi**: Kapan Anda memecah program menjadi beberapa fungsi? Berikan contoh satu program yang menurut Anda lebih baik jika di-modularisasi.

Asesmen (Evaluasi Kinerja)

Instrumen untuk Sub-CPMK 4.1: Buat program C dengan fungsi `int max3(int a, int b, int c)` yang mengembalikan nilai terbesar dari tiga bilangan. Panggil dari `main` dengan input dari user. Tambahkan fungsi `void cetakBilangan(int n)` yang menampilkan 1 sampai n.

Checklist Pencapaian Kompetensi

Centang item berikut setelah Anda yakin telah menguasainya:

☐ Saya dapat mendefinisikan dan memanggil fungsi

☐ Saya memahami parameter dan return value

☐ Saya memahami scope variabel

Rangkuman

Fungsi modularisasi program. Parameter sebagai input; return sebagai output. Variabel lokal vs global.

Bab 12

Array Satu Dimensi

Sub-CPMK yang Dicakup dalam Bab Ini:

- Sub-CPMK 4.2: Menggunakan array satu dimensi dalam program

Materi Pokok: Deklarasi dan inisialisasi array 1D; indeks dari 0; operasi dasar: input, output, penjumlahan, pencarian [3, 7].

12.1 Deklarasi dan Akses Array Satu Dimensi

Array adalah kumpulan elemen bertipe sama yang disimpan berurutan dalam memori. Deklarasi: `tipe nama[ukuran];` misalnya `int arr[10];`. Indeks dimulai dari 0; elemen terakhir berindeks ukuran-1. Akses elemen: `arr[i]`. Inisialisasi: `int arr[] = {1, 2, 3};` ukuran otomatis 3. Array tidak mengecek batas indeks; akses di luar range menyebabkan undefined behavior. Array diteruskan ke fungsi dengan nama (pointer ke elemen pertama).

12.2 Operasi Dasar pada Array

Operasi dasar: input (loop dengan scanf), output (loop dengan printf), penjumlahan elemen, pencarian nilai (linear search), mencari nilai maks/min. Contoh input: `for (i=0; i<n; i++) scanf("%d", &arr[i]);`. Ukuran array bisa konstan atau variabel (C99: Variable Length Array). Array tidak menyimpan ukuran; programmer harus melacak jumlah elemen valid. Operasi pengurutan dan pencarian akan dibahas di bab terpisah.

Aktivitas Pembelajaran

1. Buat program membaca n bilangan, menyimpan di array, lalu menampilkan rata-rata.

2. Buat program mencari nilai terbesar dalam array.

Latihan dan Refleksi

1. Mengapa indeks array dimulai dari 0?

2. Buat program membalik urutan elemen array!

3. **Refleksi**: Kesalahan apa yang sering terjadi saat mengakses array (off-by-one, out of bounds)? Bagaimana Anda menghindarinya?

Asesmen (Evaluasi Kinerja)

Instrumen untuk Sub-CPMK 4.2: Buat program C yang membaca N bilangan ke dalam array, lalu menampilkan nilai terbesar, terkecil, dan rata-ratanya. Validasi N (misalnya 1–100). Jelaskan penggunaan indeks dalam loop Anda.

Checklist Pencapaian Kompetensi

Centang item berikut setelah Anda yakin telah menguasainya:

☐ Saya dapat mendeklarasikan dan mengakses array 1D

☐ Saya dapat melakukan operasi dasar pada array

Rangkuman

Array 1D: kumpulan elemen bertipe sama. Indeks dari 0. Operasi: input, output, penjumlahan, pencarian.

Bab 13

Array Dua Dimensi dan Matriks

Sub-CPMK yang Dicakup dalam Bab Ini:

- Sub-CPMK 4.2: Menggunakan array dua dimensi dan operasi matriks

Materi Pokok: Array 2D dan representasi row-major; operasi matriks: penjumlahan, transpose, perkalian [3, 7].

13.1 Array Dua Dimensi

Array 2D merepresentasikan matriks: baris dan kolom. Deklarasi: `tipe nama[baris][kolom];` misalnya `int mat[3][4];`. Akses: `mat[i][j]` untuk baris i kolom j. Dalam memori, elemen disimpan secara row-major (baris demi baris).

Inisialisasi: `int m[2][3] = {{1,2,3},{4,5,6}};`. Loop bersarang untuk traversing: loop luar baris, loop dalam kolom. Array 2D sebagai parameter fungsi: tentukan jumlah kolom, misalnya `func(int arr[][10])`.

13.2 Operasi Matriks Sederhana

Operasi matriks dasar: penjumlahan (elemen seindeks dijumlahkan), pengurangan, transpose (baris jadi kolom, kolom jadi baris). Untuk perkalian matriks: baris A dikali kolom B; hasil matriks berukuran (baris A x kolom B). Contoh penjumlahan: C[i][j] = A[i][j] + B[i][j] untuk semua i, j. Transpose: B[j][i] = A[i][j]. Perhatikan dimensi matriks untuk operasi yang valid (penjumlahan butuh ukuran sama).

Aktivitas Pembelajaran

1. Buat program penjumlahan dua matriks.

2. Buat program transpose matriks.

Latihan dan Refleksi

1. Jelaskan representasi array 2D dalam memori (row-major)!

2. Buat program perkalian matriks 2x2!

3. **Refleksi**: Kapan Anda memilih array 2D daripada beberapa array 1D? Berikan contoh masalah yang cocok dengan array 2D.

Asesmen (Evaluasi Kinerja)

Instrumen untuk Sub-CPMK 4.2: Buat program C yang membaca dua matriks 2x2, menghitung penjumlahan dan perkalian keduanya, lalu menampilkan hasil. Gunakan nested loop untuk perkalian matriks.

Checklist Pencapaian Kompetensi

Centang item berikut setelah Anda yakin telah menguasainya:

☐ Saya dapat menggunakan array 2D

☐ Saya dapat melakukan operasi matriks sederhana

Rangkuman

Array 2D untuk matriks. Akses mat[i][j]. Operasi: penjumlahan, transpose, perkalian.

Bab 14

String dalam C

Sub-CPMK yang Dicakup dalam Bab Ini:

- Sub-CPMK 4.2: Menggunakan string dan manipulasi teks dalam C

Materi Pokok: String sebagai array `char` dengan null terminator; fungsi `strlen`, `strcpy`, `strcmp`, `strcat` dari `string.h`.[3, 5]

14.1 String dalam C

String dalam C adalah array karakter yang diakhiri null character `'\0'`. Deklarasi: `char str[50];` atau `char str[] = "Hello";`. String literal menggunakan tanda kutip ganda. Fungsi `scanf("%s", str)` membaca kata (tanpa spasi); `gets` atau `fgets` untuk baris lengkap. Header `string.h` menyediakan `strlen`, `strcpy`, `strcmp`, `strcat`. Perhatikan ukuran array cukup untuk menyimpan string termasuk `'\0'`.

14.2 Manipulasi String

Fungsi `strlen(str)` mengembalikan panjang string tanpa null. `strcpy(dest, src)` menyalin src ke dest. `strcmp(s1, s2)` membandingkan: return 0 jika sama, <0 jika s1<s2, >0 jika s1>s2. `strcat(dest, src)` menggabungkan src ke akhir dest. Pastikan buffer dest cukup besar. Untuk input aman gunakan `fgets(str, size, stdin)`. Manipulasi manual: loop melalui indeks hingga `'\0'`. Contoh: membalik string, menghitung jumlah karakter tertentu.

Aktivitas Pembelajaran

1. Buat program mengecek palindrom.

2. Buat program menghitung jumlah kata dalam kalimat.

Latihan dan Refleksi

1. Mengapa string C diakhiri '\0'?

2. Jelaskan perbedaan strcpy dan strcmp!

3. **Refleksi**: Apa risiko jika lupa mengalokasikan ruang cukup untuk string (termasuk \0) atau lupa null terminator? Berikan contoh.

Asesmen (Evaluasi Kinerja)

Instrumen untuk Sub-CPMK 4.2: Buat program C yang membaca sebuah kata, lalu mengecek apakah kata tersebut palindrom (dibaca sama dari kiri dan kanan). Gunakan loop dan bandingkan karakter, atau gunakan fungsi dari `string.h`.

Checklist Pencapaian Kompetensi

Centang item berikut setelah Anda yakin telah menguasainya:

☐ Saya dapat menggunakan string dan fungsi string.h

☐ Saya dapat memanipulasi string

Rangkuman

String = array char + null terminator. Fungsi: strlen, strcpy, strcmp, strcat.

Bab 15

Struct dan Tipe Data Bentukan

Sub-CPMK yang Dicakup dalam Bab Ini:

- Sub-CPMK 4.2: Menggunakan struct/tipe data bentukan dan array of struct dalam program

Materi Pokok: Definisi `struct`, akses anggota, `typedef`; array of struct untuk kumpulan data terstruktur [3, 7].

15.1 Definisi Struct

Struct (structure) adalah tipe data bentukan yang menggabungkan beberapa variabel dengan tipe berbeda dalam satu unit. Struct memungkinkan programmer merepresentasikan entitas dunia nyata (misalnya Mahasiswa dengan NIM, nama, IPK) dalam satu variabel terstruktur. Struct adalah salah satu fitur fundamental dalam C untuk mengorganisir data yang kompleks.

15.1.1 Deklarasi Struct

Sintaks dasar deklarasi struct:

```
struct NamaStruct {
    tipe_data field1;
    tipe_data field2;
    // ... field lainnya
};
```

Contoh dasar - Struct Mahasiswa:

```
struct Mahasiswa {
    int nim;
```

```
    char nama[50];
    float ipk;
    int umur;
};
```

15.1.2 Definisi Variabel Struct

Setelah struct dideklarasikan, kita dapat membuat variabel dari struct tersebut:

```
// Metode 1: Deklarasi terpisah
struct Mahasiswa mhs1;

// Metode 2: Deklarasi dengan inisialisasi
struct Mahasiswa mhs2 = {12345, "Budi Santoso", 3.75, 20};

// Metode 3: Menggunakan typedef (lebih praktis)
typedef struct {
    int nim;
    char nama[50];
    float ipk;
    int umur;
} Mahasiswa;

// Setelah typedef, deklarasi lebih sederhana
Mahasiswa mhs3;
```

15.1.3 Akses Anggota Struct

Anggota struct diakses menggunakan operator dot (.):

```
#include <stdio.h>
#include <string.h>

typedef struct {
    int nim;
    char nama[50];
    float ipk;
    int umur;
} Mahasiswa;

int main() {
    Mahasiswa mhs;

    // Mengisi nilai ke anggota struct
    mhs.nim = 2024001234;
    strcpy(mhs.nama, "Ahmad Fauzi");
    mhs.ipk = 3.85;
    mhs.umur = 21;

```

```
    // Mengakses dan menampilkan nilai
    printf("NIM: %d\n", mhs.nim);
    printf("Nama: %s\n", mhs.nama);
    printf("IPK: %.2f\n", mhs.ipk);
    printf("Umur: %d tahun\n", mhs.umur);

    return 0;
}
```

15.1.4 Nested Struct

Struct dapat berisi struct lain sebagai anggotanya:

```
#include <stdio.h>
#include <string.h>

// Struct untuk tanggal
typedef struct {
    int hari;
    int bulan;
    int tahun;
} Tanggal;

// Struct untuk alamat
typedef struct {
    char jalan[100];
    char kota[50];
    char provinsi[50];
    int kodePos;
} Alamat;

// Struct mahasiswa yang menggunakan struct lain
typedef struct {
    int nim;
    char nama[50];
    Tanggal tanggalLahir;
    Alamat alamat;
    float ipk;
} MahasiswaLengkap;

int main() {
    MahasiswaLengkap mhs;

    // Mengisi data mahasiswa
    mhs.nim = 2024001234;
    strcpy(mhs.nama, "Siti Nurhaliza");

    // Mengisi data tanggal lahir
    mhs.tanggalLahir.hari = 15;
    mhs.tanggalLahir.bulan = 8;
```

```
    mhs.tanggalLahir.tahun = 2002;

    // Mengisi data alamat
    strcpy(mhs.alamat.jalan, "Jl. Merdeka No. 123");
    strcpy(mhs.alamat.kota, "Jakarta");
    strcpy(mhs.alamat.provinsi, "DKI Jakarta");
    mhs.alamat.kodePos = 12345;

    mhs.ipk = 3.90;

    // Menampilkan data
    printf("Data Mahasiswa:\n");
    printf("NIM: %d\n", mhs.nim);
    printf("Nama: %s\n", mhs.nama);
    printf("Tanggal Lahir: %d/%d/%d\n",
           mhs.tanggalLahir.hari,
           mhs.tanggalLahir.bulan,
           mhs.tanggalLahir.tahun);
    printf("Alamat: %s, %s, %s %d\n",
           mhs.alamat.jalan,
           mhs.alamat.kota,
           mhs.alamat.provinsi,
           mhs.alamat.kodePos);
    printf("IPK: %.2f\n", mhs.ipk);

    return 0;
}
```

15.1.5 Array of Struct

Struct dapat dibuat dalam bentuk array untuk mengelola banyak entitas:

```
#include <stdio.h>
#include <string.h>

typedef struct {
    int nim;
    char nama[50];
    float ipk;
} Mahasiswa;

int main() {
    // Array of struct
    Mahasiswa kelas[5];
    int i;

    // Mengisi data untuk 5 mahasiswa
    for (i = 0; i < 5; i++) {
        printf("Mahasiswa ke-%d:\n", i + 1);
        printf("NIM: ");
```

```
19          scanf("%d", &kelas[i].nim);
20          printf("Nama: ");
21          scanf("%s", kelas[i].nama);
22          printf("IPK: ");
23          scanf("%f", &kelas[i].ipk);
24          printf("\n");
25      }
26
27      // Menampilkan semua data
28      printf("\nData Semua Mahasiswa:\n");
29      for (i = 0; i < 5; i++) {
30          printf("%d. %s - NIM: %d - IPK: %.2f\n",
31                  i + 1, kelas[i].nama, kelas[i].nim, kelas[i].ipk
                       );
32      }
33
34      return 0;
35  }
```

15.1.6 Keuntungan Menggunakan Struct

- **Organisasi Data:** Mengelompokkan data terkait dalam satu unit
- **Readability:** Kode lebih mudah dibaca dan dimengerti
- **Maintainability:** Modifikasi data menjadi lebih terstruktur
- **Real-world Modeling:** Mudah memodelkan entitas dunia nyata
- **Parameter Passing:** Memudahkan passing data kompleks ke fungsi

Struct adalah fondasi untuk pemrograman berorientasi objek dan sangat penting dalam pengembangan aplikasi yang mengelola data kompleks.

15.2 Array of Struct

Array dapat menyimpan elemen bertipe struct. Deklarasi: `struct Mahasiswa mhs[100];`. Akses: `mhs[i].nim`, `mhs[i].nama`. Berguna untuk mengelola data banyak entitas (daftar mahasiswa, barang, dll). Input/output dengan loop.

Struct dapat diteruskan ke fungsi pass by value atau pass by reference dengan pointer. Typedef: `typedef struct Mahasiswa Mhs;` lalu `Mhs mhs[100];`.

Aktivitas Pembelajaran

1. Buat struct Buku (judul, pengarang, tahun), lalu buat array of struct.

2. Buat program mengelola data mahasiswa dengan struct.

Latihan dan Refleksi

1. Jelaskan kegunaan struct dan kapan menggunakannya!

2. Bagaimana akses anggota struct dalam array of struct?

3. **Refleksi**: Kapan Anda memilih struct daripada beberapa array terpisah (misalnya array NIM, array nama)? Apa keuntungannya?

Asesmen (Evaluasi Kinerja)

Instrumen untuk Sub-CPMK 4.2: Buat program C dengan struct `Mahasiswa` (NIM, nama, nilai). Simpan data 3 mahasiswa dalam array of struct. Tampilkan data dan hitung rata-rata nilai. Jelaskan cara akses `arr[i].nilai`.

Checklist Pencapaian Kompetensi

Centang item berikut setelah Anda yakin telah menguasainya:

☐ Saya dapat mendefinisikan struct

☐ Saya dapat menggunakan array of struct

Rangkuman

Struct menggabungkan variabel beda tipe. Array of struct untuk data banyak entitas.

Bab 16

Algoritma Pengurutan dan Pencarian

Sub-CPMK yang Dicakup dalam Bab Ini:

- Sub-CPMK 5.1: Mengimplementasikan bubble sort, selection sort, linear search, binary search

Materi Pokok: Algoritma pengurutan (bubble sort, selection sort) dan pencarian (linear search, binary search); kompleksitas $O(n^2)$ dan $O(n)/O(\log n)$ [18, 10, 17, 12, 13, 9, 1].

16.1 Algoritma Pengurutan: Bubble Sort dan Selection Sort

Algoritma pengurutan (sorting) adalah proses mengatur elemen-elemen array dalam urutan tertentu (ascending atau descending). Dua algoritma sorting dasar yang akan dipelajari adalah Bubble Sort dan Selection Sort, keduanya memiliki kompleksitas $O(n^2)$ namun dengan pendekatan yang berbeda.

16.1.1 Bubble Sort

Bubble Sort adalah algoritma sorting sederhana yang bekerja dengan cara membandingkan elemen-elemen bertetangga dan menukarnya jika urutannya salah. Proses ini diulang hingga tidak ada lagi pertukaran yang terjadi.

Algoritma Bubble Sort:

1. Bandingkan elemen ke-i dengan elemen ke-(i+1)
2. Jika elemen ke-i > elemen ke-(i+1), tukar posisinya

3. Ulangi langkah 1-2 untuk semua elemen
4. Lakukan proses di atas hingga tidak ada pertukaran

Implementasi Bubble Sort dalam C:

```c
#include <stdio.h>

void bubbleSort(int arr[], int n) {
    int i, j, temp;
    int swapped;

    for (i = 0; i < n-1; i++) {
        swapped = 0;
        // Loop dalam untuk membandingkan elemen bertetangga
        for (j = 0; j < n-i-1; j++) {
            if (arr[j] > arr[j+1]) {
                // Tukar elemen
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
                swapped = 1;
            }
        }

        // Jika tidak ada pertukaran, array sudah terurut
        if (swapped == 0) {
            break;
        }
    }
}

void printArray(int arr[], int size) {
    int i;
    for (i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

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

    printf("Array sebelum sorting: ");
    printArray(arr, n);

    bubbleSort(arr, n);

    printf("Array setelah sorting: ");
    printArray(arr, n);

```

```
47      return 0;
48  }
```

Kode Program 16.1: Implementasi Bubble Sort

16.1.2 Selection Sort

Selection Sort adalah algoritma sorting yang bekerja dengan cara mencari elemen minimum dari sisa array yang belum terurut, lalu menempatkannya di posisi yang benar.

Algoritma Selection Sort:

1. Cari elemen minimum dari seluruh array
2. Tukar elemen minimum dengan elemen pertama
3. Cari elemen minimum dari sisa array (mulai dari elemen kedua)
4. Tukar elemen minimum dengan elemen kedua
5. Ulangi hingga seluruh array terurut

Implementasi Selection Sort dalam C:

```
1   #include <stdio.h>
2
3   void selectionSort(int arr[], int n) {
4       int i, j, min_idx, temp;
5
6       // Loop luar untuk setiap posisi
7       for (i = 0; i < n-1; i++) {
8           // Cari elemen minimum dari sisa array
9           min_idx = i;
10          for (j = i+1; j < n; j++) {
11              if (arr[j] < arr[min_idx]) {
12                  min_idx = j;
13              }
14          }
15
16          // Tukar elemen minimum dengan elemen ke-i
17          if (min_idx != i) {
18              temp = arr[i];
19              arr[i] = arr[min_idx];
20              arr[min_idx] = temp;
21          }
22      }
23  }
24
25  void printArray(int arr[], int size) {
26      int i;
27      for (i = 0; i < size; i++) {
28          printf("%d ", arr[i]);
```

```
    }
    printf("\n");
}

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

    printf("Array sebelum sorting: ");
    printArray(arr, n);

    selectionSort(arr, n);

    printf("Array setelah sorting: ");
    printArray(arr, n);

    return 0;
}
```

Kode Program 16.2: Implementasi Selection Sort

16.1.3 Analisis Kompleksitas

Algoritma	**Best Case**	**Average Case**	**Worst Case**
Bubble Sort	O(n)	$O(n^2)$	$O(n^2)$
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$

Tabel 16.1: Kompleksitas Waktu Algoritma Sorting

16.1.4 Perbandingan Bubble Sort vs Selection Sort

Aspek	**Bubble Sort**	**Selection Sort**
Prinsip Kerja	Tukar elemen bertetangga	Cari minimum, tempatkan di posisi
Jumlah Perbandingan	$O(n^2)$	$O(n^2)$
Jumlah Pertukaran	Bervariasi (0 hingga $O(n^2)$)	Selalu O(n)
Stabilitas	Stabil	Tidak stabil
Best Case	O(n) - data sudah terurut	$O(n^2)$ - selalu sama
Memory	O(1) - in-place	O(1) - in-place

Tabel 16.2: Perbandingan Bubble Sort dan Selection Sort

16.1.5 Visualisasi Proses Sorting

Contoh Bubble Sort: [5, 1, 4, 2, 8]

- Pass 1: [1, 5, 4, 2, 8] → [1, 4, 5, 2, 8] → [1, 4, 2, 5, 8] → [1, 4, 2, 5, 8]
- Pass 2: [1, 4, 2, 5, 8] → [1, 2, 4, 5, 8] → [1, 2, 4, 5, 8]
- Pass 3: [1, 2, 4, 5, 8] → [1, 2, 4, 5, 8] (sudah terurut)

Contoh Selection Sort: [5, 1, 4, 2, 8]

- Pass 1: Minimum=1 → [1, 5, 4, 2, 8]
- Pass 2: Minimum=2 → [1, 2, 4, 5, 8]
- Pass 3: Minimum=4 → [1, 2, 4, 5, 8] (sudah terurut)

Kedua algoritma ini cocok untuk data kecil ($n < 1000$) atau untuk tujuan pembelajaran konsep sorting dasar. Untuk data yang lebih besar, algoritma yang lebih efisien seperti Quick Sort atau Merge Sort lebih direkomendasikan.

16.2 Algoritma Pencarian: Linear Search dan Binary Search

Algoritma pencarian (searching) adalah proses menemukan elemen tertentu dalam kumpulan data. Dua algoritma pencarian dasar yang akan dipelajari adalah Linear Search dan Binary Search dengan karakteristik yang sangat berbeda.

16.2.1 Linear Search

Linear Search adalah algoritma pencarian paling sederhana yang bekerja dengan cara memeriksa setiap elemen secara berurutan dari awal hingga elemen yang dicari ditemukan atau sampai akhir array.

Algoritma Linear Search:

1. Mulai dari elemen pertama array
2. Bandingkan elemen saat ini dengan nilai yang dicari
3. Jika sama, kembalikan indeks elemen tersebut
4. Jika tidak, lanjut ke elemen berikutnya
5. Ulangi hingga elemen ditemukan atau akhir array
6. Jika tidak ditemukan, kembalikan -1

Implementasi Linear Search dalam C:

```c
#include <stdio.h>

int linearSearch(int arr[], int n, int x) {
    int i;
    for (i = 0; i < n; i++) {
        if (arr[i] == x) {
            return i;  // Elemen ditemukan
        }
    }
    return -1;  // Elemen tidak ditemukan
}

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

    if (result == -1) {
        printf("Elemen %d tidak ditemukan dalam array\n", x);
    } else {
        printf("Elemen %d ditemukan pada indeks %d\n", x,
           result);
    }

    return 0;
}
```

Kode Program 16.3: Implementasi Linear Search

16.2.2 Binary Search

Binary Search adalah algoritma pencarian yang sangat efisien untuk data yang sudah terurut. Algoritma ini bekerja dengan cara membagi ruang pencarian menjadi dua bagian secara berulang.

Algoritma Binary Search:

1. Tentukan batas bawah (low) dan batas atas (high) array
2. Hitung indeks tengah: mid = low + (high - low) / 2
3. Bandingkan elemen tengah dengan nilai yang dicari
4. Jika sama, kembalikan indeks tengah
5. Jika nilai dicari lebih kecil, ubah high = mid - 1
6. Jika nilai dicari lebih besar, ubah low = mid + 1
7. Ulangi hingga elemen ditemukan atau low > high

Implementasi Binary Search dalam C:

```c
#include <stdio.h>

int binarySearch(int arr[], int n, int x) {
    int low = 0, high = n - 1;

    while (low <= high) {
        int mid = low + (high - low) / 2;

        if (arr[mid] == x) {
            return mid;  // Elemen ditemukan
        }

        if (arr[mid] < x) {
            low = mid + 1;  // Cari di bagian kanan
        } else {
            high = mid - 1;  // Cari di bagian kiri
        }
    }

    return -1;  // Elemen tidak ditemukan
}

int main() {
    int arr[] = {11, 12, 22, 25, 34, 64, 90};  // Array harus terurut
    int n = sizeof(arr)/sizeof(arr[0]);
    int x = 22;
    int result = binarySearch(arr, n, x);

    if (result == -1) {
        printf("Elemen %d tidak ditemukan dalam array\n", x);
    } else {
        printf("Elemen %d ditemukan pada indeks %d\n", x,
           result);
    }

    return 0;
}
```

Kode Program 16.4: Implementasi Binary Search

16.2.3 Analisis Kompleksitas Pencarian

Algoritma	Best Case	Average Case	Worst Case
Linear Search	O(1)	O(n)	O(n)
Binary Search	O(1)	O(log n)	O(log n)

Tabel 16.3: Kompleksitas Waktu Algoritma Pencarian

16.2.4 Perbandingan Linear Search vs Binary Search

Aspek	**Linear Search**	**Binary Search**
Prinsip Kerja	Periksa elemen satu per satu	Bagi dua ruang pencarian
Syarat Data	Tidak perlu terurut	Harus terurut
Kompleksitas Waktu	O(n)	O(log n)
Kompleksitas Ruang	O(1)	O(1)
Best Case	O(1) - elemen pertama	O(1) - elemen tengah
Worst Case	O(n) - elemen terakhir	O(log n)
Implementasi	Sangat sederhana	Lebih kompleks

Tabel 16.4: Perbandingan Linear Search dan Binary Search

16.2.5 Visualisasi Proses Pencarian

Contoh Linear Search: Mencari 25 dalam [64, 34, 25, 12, 22]

- Langkah 1: Bandingkan 64 dengan 25 → tidak sama
- Langkah 2: Bandingkan 34 dengan 25 → tidak sama
- Langkah 3: Bandingkan 25 dengan 25 → sama! Ditemukan di indeks 2

Contoh Binary Search: Mencari 25 dalam [11, 12, 22, 25, 34, 64, 90]

- Langkah 1: low=0, high=6, mid=3 → arr[3]=25 → sama! Ditemukan di indeks 3

Contoh Binary Search: Mencari 22 dalam [11, 12, 22, 25, 34, 64, 90]

- Langkah 1: low=0, high=6, mid=3 → arr[3]=25 > 22 → high=2
- Langkah 2: low=0, high=2, mid=1 → arr[1]=12 < 22 → low=2
- Langkah 3: low=2, high=2, mid=2 → arr[2]=22 → sama! Ditemukan di indeks 2

16.2.6 Kapan Menggunakan Algoritma Pencarian

- **Gunakan Linear Search jika:**
  - Data tidak terurut atau sering berubah
  - Jumlah data kecil (n < 100)
  - Implementasi yang sederhana diutamakan
  - Hanya satu kali pencarian
- **Gunakan Binary Search jika:**

- Data sudah terurut atau bisa diurutkan terlebih dahulu
- Jumlah data besar (n > 1000)
- Pencarian dilakukan berulang kali
- Performa pencarian sangat penting

Pemilihan algoritma pencarian yang tepat sangat mempengaruhi efisiensi program, terutama untuk aplikasi yang menangani data dalam jumlah besar.

16.3 Analisis Big O dan Visualisasi Kompleksitas

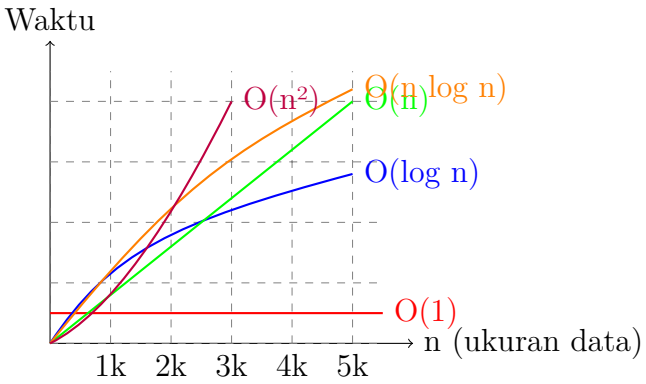
Big O notation adalah notasi matematika yang digunakan untuk menggambarkan kompleksitas waktu atau ruang suatu algoritma. Notasi ini membantu kita memahami bagaimana performa algoritma berubah seiring dengan pertambahan ukuran input.

16.3.1 Konsep Dasar Big O Notation

Big O notation menggambarkan batas atas (worst case) dari pertumbuhan fungsi kompleksitas algoritma. Beberapa notasi Big O yang umum:

- **O(1)** - Konstan: Waktu eksekusi tidak bergantung pada ukuran input
- **O(log n)** - Logaritmik: Waktu eksekusi tumbuh secara logaritmik
- **O(n)** - Linear: Waktu eksekusi sebanding dengan ukuran input
- **O(n log n)** - Linearitmaik: Kombinasi linear dan logaritmik
- **O(n²)** - Kuadratik: Waktu eksekusi sebanding dengan kuadrat ukuran input

16.3.2 Grafik Kompleksitas Algoritma



Algoritma	Best Case	Average	Worst Case	Space
Bubble Sort	O(n)	$O(n^2)$	$O(n^2)$	O(1)
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	O(1)
Linear Search	O(1)	O(n)	O(n)	O(1)
Binary Search	O(1)	O(log n)	O(log n)	O(1)

Tabel 16.5: Tabel Kompleksitas Lengkap Algoritma

n	O(log n)	O(n)	O(n log n)	$O(n^2)$
10	3	10	33	100
100	7	100	664	10.000
1.000	10	1.000	9.966	1.000.000
10.000	13	10.000	132.877	100.000.000
100.000	17	100.000	1.660.964	10.000.000.000

Tabel 16.6: Perbandingan Jumlah Operasi untuk Berbagai Kompleksitas

16.3.3 Tabel Kompleksitas Lengkap

16.3.4 Analisis Performa Praktis

16.3.5 Implementasi dengan Pengukuran Waktu

Berikut implementasi lengkap dengan pengukuran waktu eksekusi untuk membandingkan performa algoritma:

```c
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

// Fungsi untuk mengukur waktu eksekusi
double measureTime(clock_t start, clock_t end) {
    return ((double)(end - start)) / CLOCKS_PER_SEC;
}

// Bubble Sort dengan pengukuran waktu
void bubbleSortWithTiming(int arr[], int n) {
    clock_t start = clock();

    int i, j, temp, swapped;
    for (i = 0; i < n-1; i++) {
        swapped = 0;
        for (j = 0; j < n-i-1; j++) {
            if (arr[j] > arr[j+1]) {
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
                swapped = 1;
            }
        }
```

```
25          if (swapped == 0) break;
26      }
27
28      clock_t end = clock();
29      printf("Bubble Sort waktu: %.6f detik\n", measureTime(start
            , end));
30  }
31
32  // Binary Search dengan pengukuran waktu
33  int binarySearchWithTiming(int arr[], int n, int x) {
34      clock_t start = clock();
35
36      int low = 0, high = n - 1;
37      while (low <= high) {
38          int mid = low + (high - low) / 2;
39          if (arr[mid] == x) {
40              clock_t end = clock();
41              printf("Binary Search waktu: %.6f detik\n",
                    measureTime(start, end));
42              return mid;
43          }
44          if (arr[mid] < x) low = mid + 1;
45          else high = mid - 1;
46      }
47
48      clock_t end = clock();
49      printf("Binary Search waktu: %.6f detik\n", measureTime(
            start, end));
50      return -1;
51  }
52
53  int main() {
54      const int SIZE = 10000;
55      int arr[SIZE];
56
57      // Generate random data
58      srand(time(NULL));
59      for (int i = 0; i < SIZE; i++) {
60          arr[i] = rand() % 10000;
61      }
62
63      printf("Mengurutkan %d elemen...\n", SIZE);
64      bubbleSortWithTiming(arr, SIZE);
65
66      printf("Mencari elemen...\n");
67      int result = binarySearchWithTiming(arr, SIZE, 5000);
68
69      if (result != -1) {
70          printf("Elemen ditemukan pada indeks %d\n", result);
71      } else {
72          printf("Elemen tidak ditemukan\n");
```

```
    }

    return 0;
}
```

Kode Program 16.5: Program Perbandingan Performa Algoritma

16.3.6 Tips Optimasi Algoritma

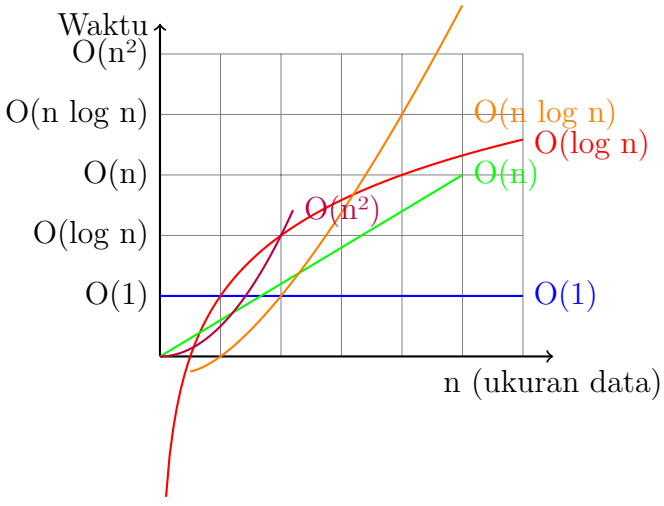
- **Pilih algoritma yang tepat:** Sesuaikan dengan karakteristik data
- **Pertimbangkan ukuran data:** Untuk data kecil, algoritma sederhana mungkin lebih cepat
- **Analisis trade-off:** Memory vs speed, simplicity vs efficiency
- **Profile kode:** Ukur performa aktual untuk identifikasi bottleneck
- **Gunakan algoritma yang sudah teruji:** Library function biasanya sudah dioptimasi

Pemahaman Big O notation sangat penting untuk mengembangkan algoritma yang efisien dan scalable, terutama untuk aplikasi yang menangani data dalam jumlah besar.

16.4 Grafik Kompleksitas dan Visualisasi

Memahami kompleksitas algoritma sangat penting untuk memilih algoritma yang tepat untuk masalah tertentu. Grafik kompleksitas membantu visualisasi performa algoritma pada berbagai ukuran input.

16.4.1 Grafik Kompleksitas Waktu



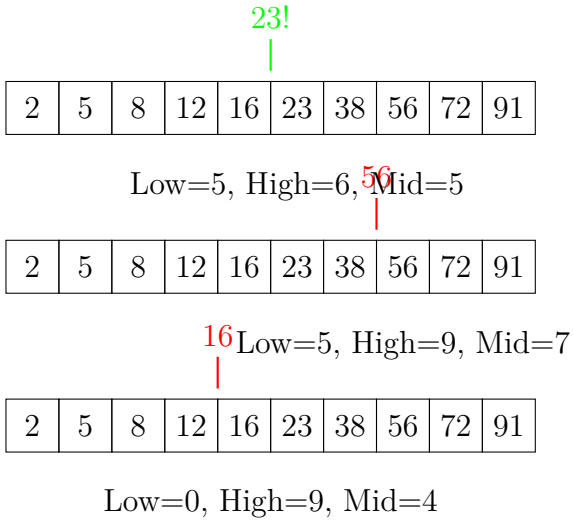
16.4.2 Tabel Perbandingan Komprehensif

Algoritma	**Waktu**	**Space**	**Stabil**	**Kapan Digunakan**
Bubble Sort	$O(n^2)$	O(1)	Ya	Data kecil, sudah hampir terurut
Selection Sort	$O(n^2)$	O(1)	Tidak	Data kecil, meminimalkan pertukaran
Linear Search	O(n)	O(1)	Ya	Data tidak terurut, pencarian tunggal
Binary Search	O(log n)	O(1)	Ya	Data terurut, pencarian berulang

Tabel 16.7: Perbandingan Lengkap Algoritma Sorting dan Searching

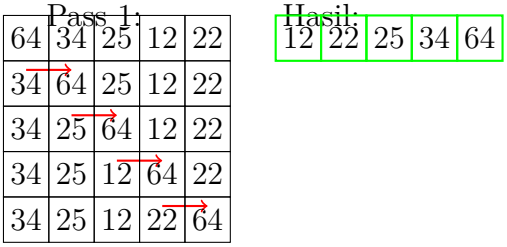
16.4.3 Visualisasi Proses Binary Search

Contoh Pencarian 23 dalam [2, 5, 8, 12, 16, 23, 38, 56, 72, 91]



16.4.4 Animasi Bubble Sort

Proses Bubble Sort pada [64, 34, 25, 12, 22]:



16.4.5 Analisis Performa Real-world

Ukuran Data	**Bubble Sort**	**Selection Sort**	**Binary Search**
10 elemen	0.001 ms	0.001 ms	0.0001 ms
100 elemen	0.1 ms	0.1 ms	0.0002 ms
1,000 elemen	10 ms	10 ms	0.0003 ms
10,000 elemen	1000 ms	1000 ms	0.0004 ms
100,000 elemen	100,000 ms	100,000 ms	0.0005 ms

Tabel 16.8: Estimasi Waktu Eksekusi (asumsi 1 operasi = 1 µs)

Visualisasi ini membantu mahasiswa memahami mengapa pemilihan algoritma sangat penting dalam pengembangan perangkat lunak yang efisien.

Aktivitas Pembelajaran

1. Implementasikan bubble sort dan uji dengan array acak.

2. Implementasikan binary search; pastikan array terurut terlebih dahulu.

Latihan dan Refleksi

1. Jelaskan perbedaan bubble sort dan selection sort!

2. Mengapa binary search membutuhkan data terurut? Bandingkan kompleksitas linear vs binary search.

3. **Refleksi**: Kapan Anda memilih linear search dan kapan binary search? Bagaimana jika data belum terurut?

Asesmen (Evaluasi Kinerja)

Instrumen untuk Sub-CPMK 5.1: Buat program C yang: (1) mengisi array dengan

10 bilangan acak, (2) mengurutkannya dengan selection sort dan menampilkan hasil, (3) membaca sebuah bilangan dan mencari dengan binary search lalu menampilkan indeks atau pesan tidak ditemukan.

Checklist Pencapaian Kompetensi

Centang item berikut setelah Anda yakin telah menguasainya:

☐ Saya dapat mengimplementasikan bubble sort dan selection sort

☐ Saya dapat mengimplementasikan linear search dan binary search

Rangkuman

Bubble sort dan selection sort $O(n^2)$. Linear search $O(n)$; binary search $O(\log n)$ untuk data terurut. Pemilihan algoritma bergantung pada ukuran data dan karakteristik input.

Bab 17

Evaluasi, Refleksi, dan Integrasi Kompetensi

17.1 Panduan Proyek Integrasi (Pertemuan 15)

Sesuai RPS, pertemuan 15 ditujukan untuk integrasi materi, proyek, dan review. Proyek integrasi mengukur pencapaian CPMK-2 hingga CPMK-5 dengan menggabungkan semua konsep yang telah dipelajari: algoritma, flowchart, pseudocode, variabel, struktur kontrol, fungsi, array, struct, pengurutan, dan pencarian.

17.1.1 Studi Kasus 1: Manajemen Data Mahasiswa

Deskripsi: Buat program yang mengelola data mahasiswa dengan fitur lengkap.

Spesifikasi Requirements:

- **Data Mahasiswa:** NIM (int), nama (string), IPK (float), umur (int)
- **Fitur Input:** Tambah data mahasiswa baru
- **Fitur Tampil:** Tampilkan semua data mahasiswa
- **Fitur Pencarian:** Cari mahasiswa berdasarkan NIM
- **Fitur Analisis:** Hitung rata-rata IPK semua mahasiswa
- **Fitur Pengurutan:** Urutkan data berdasarkan nama atau IPK

Teknis Implementasi:

- Gunakan `struct Mahasiswa` untuk merepresentasikan data
- Gunakan array `Mahasiswa kelas[50]` untuk menyimpan data
- Buat fungsi terpisah untuk setiap operasi
- Implementasikan menu interaktif dengan `switch-case`

- Gunakan bubble sort atau selection sort untuk pengurutan
- Gunakan linear search untuk pencarian

Contoh Struktur Program:

```c
#include <stdio.h>
#include <string.h>

typedef struct {
    int nim;
    char nama[50];
    float ipk;
    int umur;
} Mahasiswa;

// Fungsi-fungsi yang dibutuhkan
void tambahMahasiswa(Mahasiswa kelas[], int *jumlah);
void tampilkanSemua(Mahasiswa kelas[], int jumlah);
void cariMahasiswa(Mahasiswa kelas[], int jumlah);
void hitungRataIPK(Mahasiswa kelas[], int jumlah);
void urutkanBerdasarkanIPK(Mahasiswa kelas[], int jumlah);
void urutkanBerdasarkanNama(Mahasiswa kelas[], int jumlah);

int main() {
    Mahasiswa kelas[50];
    int jumlah = 0;
    int pilihan;

    do {
        // Menu utama
        printf("\n=== MENU MANAJEMEN MAHASISWA ===\n");
        printf("1. Tambah Mahasiswa\n");
        printf("2. Tampilkan Semua Mahasiswa\n");
        printf("3. Cari Mahasiswa\n");
        printf("4. Hitung Rata-rata IPK\n");
        printf("5. Urutkan berdasarkan IPK\n");
        printf("6. Urutkan berdasarkan Nama\n");
        printf("0. Keluar\n");
        printf("Pilihan: ");
        scanf("%d", &pilihan);

        switch(pilihan) {
            case 1: tambahMahasiswa(kelas, &jumlah); break;
            case 2: tampilkanSemua(kelas, jumlah); break;
            case 3: cariMahasiswa(kelas, jumlah); break;
            case 4: hitungRataIPK(kelas, jumlah); break;
            case 5: urutkanBerdasarkanIPK(kelas, jumlah); break
                ;
            case 6: urutkanBerdasarkanNama(kelas, jumlah);
                break;
        }
```

```
    } while (pilihan != 0);

    return 0;
}
```

17.1.2 Studi Kasus 2: Sistem Inventory Barang

Deskripsi: Buat program inventory untuk mengelola data barang.

Spesifikasi Requirements:

- **Data Barang:** Kode (int), nama (string), jumlah (int), harga (float)
- **Fitur CRUD:** Create, Read, Update, Delete barang
- **Fitur Pencarian:** Cari berdasarkan kode atau nama
- **Fitur Laporan:** Hitung total nilai inventory
- **Fitur Sorting:** Urutkan berdasarkan nama atau harga

17.1.3 Rubrik Penilaian Proyek

17.1.4 Tahap Pengembangan Proyek

1. **Analisis (Pekan 1):** Pahami requirements, buat flowchart
2. **Desain (Pekan 2):** Buat pseudocode, definisikan struct dan fungsi
3. **Implementasi (Pekan 3-4):** Koding fitur-fitur utama
4. **Integrasi (Pekan 5):** Gabungkan semua fitur, testing
5. **Dokumentasi (Pekan 6):** Buat laporan dan presentasi

17.1.5 Format Pengumpulan

- **Source Code:** File .c dan .h (jika ada)
- **Executable:** File .exe (hasil kompilasi)
- **Dokumentasi:** Laporan PDF (maks 10 halaman)
- **Flowchart:** Gambar flowchart (PNG/JPG)
- **Demo:** Video demo program (opsional, nilai tambah)

Kriteria	**Bobot**	**Deskripsi Penilaian**
Desain Algoritma	20%	• Flowchart lengkap dan benar (10%) • Pseudocode terstruktur (10%)
Implementasi Kode	40%	• Kode benar dan kompilasi sukses (15%) • Modularitas dan fungsi yang baik (15%) • Error handling yang memadai (10%)
Struktur Data	25%	• Penggunaan struct yang tepat (10%) • Pengelolaan array dengan benar (10%) • Manipulasi string yang proper (5%)
Integrasi Algoritma	15%	• Implementasi sorting yang benar (8%) • Implementasi searching yang efisien (7%)

Tabel 17.1: Rubrik Penilaian Proyek Integrasi

17.2 Asesmen Akhir Komprehensif

Bab ini menyajikan evaluasi komprehensif untuk mengukur pencapaian semua CPMK mata kuliah Algoritma dan Pemrograman. Asesmen meliputi kuis teori (algoritma, flowchart, pseudocode, konsep C), praktik coding (variabel, percabangan, perulangan, fungsi, array, struct), serta implementasi algoritma pengurutan dan pencarian. Mahasiswa diharapkan dapat menyelesaikan soal UTS dan UAS sesuai format yang ditetapkan dalam RPS. Rubrik penilaian mengacu pada Lampiran A.

17.2.1 Soal UTS: Teori dan Implementasi Dasar

Bagian A: Teori Algoritma (30 points)

1. Jelaskan keenam karakteristik algoritma yang baik. Berikan contoh algoritma dari kehidupan sehari-hari dan analisis apakah memenuhi semua karakteristik tersebut. (10 points)
2. Bandingkan kelebihan dan kekurangan antara flowchart dan pseudocode sebagai alat bantu perancangan algoritma. Berikan contoh kasus di mana flowchart lebih cocok dan contoh kasus di mana pseudocode lebih cocok. (10 points)
3. Analisis kompleksitas Big O dari algoritma berikut:

```c
for (int i = 0; i < n; i++) {
    for (int j = i; j < n; j++) {
        printf("%d ", arr[i] + arr[j]);
    }
}
```

   Jelaskan langkah-langkah analisisnya. (10 points)

Bagian B: Implementasi Dasar (40 points)

1. Buat program C untuk menghitung nilai akhir mahasiswa dengan ketentuan:
   - Input: nilai tugas, nilai uts, nilai uas (0-100)
   - Bobot: tugas 30%, uts 35%, uas 35%
   - Output: nilai akhir dan grade (A: 80-100, B: 70-79, C: 60-69, D: 50-59, E: <50)
   - Validasi: input harus antara 0-100

   (20 points)
2. Buat program C untuk menampilkan pola segitiga berikut:

```
*
**
***
****
*****
```

Tinggi segitiga diinput oleh user. Gunakan nested loop. (10 points)

3. Buat fungsi untuk mengecek apakah suatu bilangan adalah bilangan prima. Fungsi menerima parameter integer dan mengembalikan 1 jika prima, 0 jika bukan. (10 points)

Bagian C: Algoritma Lanjut (30 points)

1. Implementasikan algoritma selection sort untuk mengurutkan array of integers. Tampilkan proses sorting setiap iterasi. (15 points)
2. Implementasikan binary search untuk mencari elemen dalam array yang sudah terurut. Hitung jumlah perbandingan yang dilakukan. (15 points)

17.2.2 Soal UAS: Integrasi Komprehensif

Studi Kasus: Sistem Perpustakaan Mini

Buat program manajemen perpustakaan dengan spesifikasi berikut:

Data Buku:

- Kode buku (int)
- Judul (string)
- Pengarang (string)
- Tahun terbit (int)
- Status (tersedia/dipinjam)

Fitur yang Dibutuhkan:

1. **Menu Utama** dengan pilihan:
    - Tambah buku baru
    - Tampilkan semua buku
    - Cari buku berdasarkan kode
    - Pinjam buku
    - Kembalikan buku
    - Urutkan buku berdasarkan judul
    - Statistik perpustakaan
    - Keluar
2. **Validasi Input:**
    - Kode buku harus unik
    - Tahun terbit harus reasonable (1900-2024)

- Judul dan pengarang tidak boleh kosong

3. **Fitur Pencarian:** Gunakan binary search untuk pencarian berdasarkan kode (array harus diurutkan terlebih dahulu)
4. **Fitur Pengurutan:** Implementasikan bubble sort untuk mengurutkan berdasarkan judul buku
5. **Statistik:** Tampilkan total buku, buku tersedia, buku dipinjam

Kriteria Penilaian:

Aspek	**Bobot**	**Kriteria**
Struktur Data	20%	Penggunaan struct yang tepat, array management
Modularitas	25%	Fungsi yang terstruktur, parameter passing
Algoritma	20%	Implementasi sorting dan searching yang benar
User Interface	15%	Menu yang jelas, input validation
Error Handling	10%	Penanganan error yang memadai
Dokumentasi	10%	Komentar, variable naming, formatting

Tabel 17.2: Rubrik Penilaian UAS

17.2.3 Format Pengumpulan

- **Source Code:** File .c yang sudah kompilasi sukses
- **Dokumentasi:** Penjelasan algoritma dan struktur program (maks 5 halaman)
- **Screenshot:** Output program untuk setiap fitur
- **Video Demo:** Video singkat (2-3 menit) menjelaskan program (opsional, nilai tambah)

17.2.4 Kriteria Kelulusan

- **A (80-100):** Semua fitur berjalan sempurna, kode terstruktur, dokumentasi lengkap
- **B (70-79):** Sebagian besar fitur berjalan, ada beberapa bug minor
- **C (60-69):** Fitur utama berjalan, struktur dasar benar
- **D (50-59):** Beberapa fitur utama berjalan, ada bug signifikan
- **E (<50):** Program tidak berjalan atau fitur utama tidak ada

17.2.5 Pemetaan Soal ke CPMK

Tabel berikut memetakan bagian UTS dan UAS ke Capaian Pembelajaran Mata Kuliah (CPMK) agar asesmen dapat dijadikan bukti pencapaian kompetensi.

Asesmen	**CPMK**	**Keterangan**
UTS Bagian A (Teori)	CPMK-1	Karakteristik algoritma, flowchart, pseudocode, kompleksitas
UTS Bagian B (Implementasi Dasar)	CPMK-2, CPMK-3	Rancang dan implementasi: variabel, percabangan, perulangan, fungsi
UTS Bagian C (Algoritma Lanjut)	CPMK-5	Selection sort, binary search
UAS (Sistem Perpustakaan)	CPMK-2 s.d. CPMK-5	Integrasi: flowchart/pseudocode, C, struct, array, string, sorting, searching

Tabel 17.3: Pemetaan Asesmen UTS/UAS ke CPMK

17.3 Rubrik Penilaian Komprehensif

Rubrik penilaian komprehensif berbasis OBE mencakup empat aspek utama:

- **Ketepatan algoritma dan logika** (30%): Flowchart/pseudocode benar, langkah terstruktur, sesuai CPMK-1 dan CPMK-2.
- **Kebenaran implementasi kode C** (30%): Kode kompilasi sukses, output benar, sesuai CPMK-3.
- **Modularitas dan penggunaan fungsi** (20%): Fungsi dan struktur data digunakan dengan tepat, sesuai CPMK-4.
- **Keterbacaan kode dan dokumentasi** (20%): Penamaan jelas, indentasi rapi, komentar dan laporan memadai.

Nilai huruf: A (85–100): menguasai semua CPMK; B (70–84): sebagian besar CPMK; C (60–69): CPMK dasar; D (50–59): sebagian kecil; E (<50): belum menguasai. Lihat Lampiran A untuk rubrik rinci tugas praktik dan Lampiran H untuk pemetaan rubrik ke CPMK.

17.4 Tinjauan Pencapaian Kompetensi

Tinjauan ini membantu Anda memastikan kesiapan sebelum UTS/UAS. Pastikan Anda menguasai kompetensi berikut sesuai CPMK:

- **CPMK-1:** Konsep algoritma dan enam karakteristiknya; definisi dan fungsi flowchart serta pseudocode.
- **CPMK-2:** Merancang algoritma dengan flowchart dan pseudocode untuk masalah sederhana.
- **CPMK-3:** Variabel, tipe data, I/O (`printf/scanf`); operator; percabangan if-else dan switch; perulangan for, while, do-while.
- **CPMK-4:** Fungsi (parameter, return, scope); array 1D dan 2D; string dan `string.h`; struct serta array of struct.
- **CPMK-5:** Bubble sort, selection sort, linear search, binary search; pemahaman kompleksitas sederhana.

Gunakan **checklist** di setiap bab untuk self-assessment. Jika ada indikator yang belum Anda centang, pelajari kembali materi bab terkait dan kerjakan latihan. Konsultasikan dengan dosen jika ada CPMK yang belum tercapai.

17.5 Rekomendasi Tindak Lanjut

Untuk penguasaan lebih lanjut: pelajari pemrograman berorientasi objek (Java/C++) sebagai mata kuliah lanjutan; eksplorasi struktur data (linked list, stack, queue, tree); praktik di platform HackerRank, LeetCode, atau SPOJ; baca “C How to Program” (Deitel) dan “The C Programming Language” (Kernighan & Ritchie). Untuk yang masih perlu perbaikan: review bab terkait, praktik coding lebih banyak, konsultasi dengan dosen, manfaatkan sumber online (SPADA, cppreference, tutorial C).

Rangkuman

Bab ini menyajikan evaluasi komprehensif, rubrik penilaian, tinjauan capaian CPMK, dan rekomendasi tindak lanjut pembelajaran Algoritma dan Pemrograman.

Lampiran

Lampiran A: Rubrik Penilaian Tugas Praktik

Kriteria	Sangat Baik	Baik	Perlu Perbaikan
Desain Algoritma	Flowchart/pseudo-code jelas, logika benar	Desain cukup jelas	Desain belum tepat
Implementasi Kode C	Kode benar, kompilasi sukses	Ada minor error	Banyak kesalahan
Modularitas	Fungsi digunakan dengan tepat	Sebagian modular	Monolitik
Keterbacaan Kode	Nama variabel jelas, indentasi rapi	Cukup terbaca	Sulit dibaca
Ketepatan Output	Output sesuai spesifikasi	Minor deviation	Output salah

Tabel 17.4: Rubrik Penilaian Tugas Praktik Algoritma dan Pemrograman

Pemetaan Rubrik ke CPMK

Kriteria pada rubrik di atas terhubung ke CPMK sebagai berikut:

Kriteria Rubrik	CPMK	Indikator
Desain Algoritma	CPMK-1, CPMK-2	Flowchart/pseudocode jelas dan benar
Implementasi Kode C	CPMK-3	Variabel, I/O, percabangan, perulangan benar
Modularitas	CPMK-4	Penggunaan fungsi dan struktur data
Keterbacaan Kode	CPMK-3, CPMK-4	Penamaan, indentasi, dokumentasi
Ketepatan Output	CPMK-2 s.d. CPMK-5	Hasil sesuai spesifikasi dan algoritma

Tabel 17.5: Pemetaan Kriteria Rubrik ke CPMK

Lampiran B: Contoh Template Laporan Tugas

1. **Judul dan Identitas Mahasiswa**
    - Nama lengkap dan NIM
    - Kelas dan kelompok (jika ada)
    - Judul tugas/proyek
2. **Deskripsi Masalah**
    - Ringkasan requirements
    - Input dan output yang diharapkan
    - Batasan dan asumsi
3. **Flowchart dan/atau Pseudocode**
    - Desain algoritma lengkap
    - Penjelasan setiap langkah
4. **Implementasi (cuplikan kode C penting)**
    - Struktur data yang digunakan
    - Fungsi-fungsi utama
    - Bagian kode yang krusial
5. **Hasil Pengujian**
    - Screenshot output program
    - Data uji yang digunakan
    - Analisis hasil pengujian
6. **Refleksi dan Kesimpulan**
    - Pembelajaran yang didapat
    - Kesulitan yang dihadapi
    - Saran perbaikan

Lampiran H: Contoh Rubrik Refleksi dan Pemetaan ke CPMK

Contoh Rubrik Refleksi (per Bab)

Refleksi mahasiswa per bab dapat dinilai dengan rubrik berikut. Rubrik ini memetakan indikator ke CPMK yang relevan.

Kriteria	Baik (3)	Cukup (2)	Kurang (1)
Pemahaman konsep	Menjelaskan dengan tepat dan contoh jelas	Menjelaskan dengan benar, contoh kurang	Pemahaman samar atau salah
Keterkaitan dengan CPMK	Menyebutkan Sub-CPMK yang tercapai	Menyebutkan tanpa rincian	Tidak mengaitkan ke CPMK
Kesulitan dan solusi	Mengidentifikasi kesulitan dan langkah perbaikan	Mengidentifikasi kesulitan saja	Tidak reflektif

Tabel 17.6: Rubrik Refleksi per Bab

Contoh Portofolio dan Pemetaan ke CPMK

Portofolio dapat berisi: (1) hasil tugas per bab (flowchart, kode C, laporan), (2) refleksi per bab, (3) proyek integrasi. Pemetaan artefak ke CPMK:

- **CPMK-1:** Ringkasan konsep algoritma, flowchart, pseudocode; jawaban latihan teori Bab 2–4.
- **CPMK-2:** Flowchart dan pseudocode dari tugas Bab 3–4 dan proyek.
- **CPMK-3:** Kode C untuk variabel, operator, percabangan, perulangan (Bab 5–10).
- **CPMK-4:** Kode C untuk fungsi, array, string, struct (Bab 11–15).
- **CPMK-5:** Implementasi sorting dan searching (Bab 16) serta integrasi dalam proyek.

Lampiran C: Glosarium Istilah Algoritma dan C

- **Algoritma**: Urutan langkah logis dan terdefinisi untuk menyelesaikan masalah
- **Flowchart**: Representasi visual algoritma dengan simbol standar
- **Pseudocode**: Deskripsi algoritma dalam bahasa manusia menyerupai kode
- **Variabel**: Tempat penyimpanan data di memori dengan nama dan tipe
- **Tipe Data**: Klasifikasi data (int, float, char, dll)
- **Array**: Kumpulan elemen bertipe sama yang disimpan berurutan
- **Struct**: Tipe data bentukan yang menggabungkan beberapa variabel
- **Fungsi**: Blok kode yang dapat dipanggil berulang dengan nama
- **Percabangan**: Pengambilan keputusan berdasarkan kondisi (if, switch)
- **Perulangan**: Pengulangan eksekusi blok kode (for, while, do-while)

- **Sorting**: Proses mengurutkan elemen array
- **Searching**: Proses mencari elemen dalam array
- **Kompleksitas**: Ukuran efisiensi algoritma (Big O notation)
- **OBE**: Outcome-Based Education, pendidikan berbasis capaian
- **CPL**: Capaian Pembelajaran Lulusan
- **CPMK**: Capaian Pembelajaran Mata Kuliah

Lampiran D: Cheat Sheet Sintaks Bahasa C

Tipe Data Dasar

Tipe	Format	Ukuran
int	%d, %i	4 byte
float	%f	4 byte
double	%lf	8 byte
char	%c	1 byte
string (char[])	%s	-

Input/Output

```c
// Output
printf("Teks: %d", variabel_int);
printf("Float: %.2f", variabel_float);
printf("Karakter: %c", variabel_char);

// Input
scanf("%d", &variabel_int);
scanf("%f", &variabel_float);
scanf("%s", variabel_string);
scanf(" %c", &variabel_char); // spasi untuk skip whitespace
```

Struktur Kontrol

```c
// If-Else
if (kondisi) {
    // blok jika benar
} else if (kondisi_lain) {
    // blok jika kondisi lain benar
} else {
    // blok jika semua salah
}

```

```
// Switch-Case
switch (variabel) {
    case nilai1:
        // aksi untuk nilai1
        break;
    case nilai2:
        // aksi untuk nilai2
        break;
    default:
        // aksi default
        break;
}

// For Loop
for (inisialisasi; kondisi; increment) {
    // blok yang diulang
}

// While Loop
while (kondisi) {
    // blok yang diulang
}

// Do-While Loop
do {
    // blok yang diulang
} while (kondisi);
```

Array dan Struct

```
// Array
int angka[10];                    // deklarasi
angka[0] = 100;                   // assignment
int nilai[] = {90, 85, 78};       // inisialisasi

// Struct
typedef struct {
    int nim;
    char nama[50];
    float ipk;
} Mahasiswa;

Mahasiswa mhs1;                   // deklarasi
mhs1.nim = 12345;                 // assignment
strcpy(mhs1.nama, "Budi");        // string assignment
```

Lampiran E: Modul Praktikum 1 – Dasar Pemrograman

Praktikum 1 dilaksanakan pada pertemuan 7 sesuai RPS. Tujuan: menguasai Sub-CPMK 3.1 dan 3.2 (struktur seleksi dan perulangan). Lakukan aktivitas berikut secara berurutan:

1. **Bab III**: Buat flowchart untuk menentukan bilangan ganjil/genap dan konversi suhu Celsius ke Fahrenheit. Implementasikan ke program C.
2. **Bab IV**: Tulis pseudocode untuk rata-rata tiga bilangan, konversikan ke C.
3. **Bab V**: Buat program membaca nama dan umur; program konversi suhu Celsius ke Fahrenheit.
4. **Bab VI**: Buat program kalkulator sederhana dengan operator +, -, *, /.
5. **Bab VII**: Buat program menentukan bilangan ganjil/genap dan grade nilai (A–E) dari input skor.

Susun laporan menggunakan template Lampiran B. Asisten laboratorium akan menilai berdasarkan rubrik Lampiran A.

Lampiran F: Referensi Tambahan

Online Resources

- **cppreference.com**: https://en.cppreference.com/w/c - Referensi lengkap bahasa C
- **GeeksforGeeks**: https://www.geeksforgeeks.org/c/ - Tutorial dan contoh kode
- **SPADA Indonesia**: https://lmsspada.kemdiktisaintek.go.id/ - Materi pembelajaran resmi
- **GCC Documentation**: https://gcc.gnu.org/onlinedocs/ - Dokumentasi compiler
- **Learn-C.org**: https://www.learn-c.org/ - Tutorial interaktif C

Buku Referensi

- Kernighan, B.W., & Ritchie, D.M. (1988). *The C Programming Language*. Prentice Hall.
- Deitel, H.M., & Deitel, P.J. (2016). *C How to Program*. Pearson.
- Cormen, T.H., et al. (2009). *Introduction to Algorithms*. MIT Press.

Alat Pengembangan

- **Compiler**: GCC/MinGW, Clang, Visual Studio C++
- **IDE**: Code::Blocks, Dev-C++, Visual Studio Code
- **Debugger**: GDB, integrated debugger dalam IDE
- **Version Control**: Git, GitHub Desktop

Lampiran G: Soal Latihan Tambahan

Level Dasar

1. Buat program menghitung luas persegi panjang
2. Buat program konversi kilometer ke mil
3. Buat program menampilkan bilangan genap 1-100

Level Menengah

1. Buat program kalkulator scientific (+, -, *, /, %, pangkat)
2. Buat program menentukan tahun kabisat
3. Buat program menghitung faktorial

Level Lanjutan

1. Implementasi insertion sort dan merge sort
2. Buat program game tebak angka
3. Buat program database sederhana dengan file I/O

Daftar Pustaka

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 3rd edition, 2009.

[2] C operator precedence. https://en.cppreference.com/w/c/language/operator_precedence, 2024. Accessed 2026-02-12.

[3] C reference. https://en.cppreference.com/w/c, 2024.

[4] Control flow. https://en.cppreference.com/w/c/language/statements, 2024. C reference - statements and control flow.

[5] Null-terminated byte strings. https://en.cppreference.com/w/c/string/byte, 2024. C reference - string.h.

[6] Introduction to algorithms. https://www.geeksforgeeks.org/introduction-to-algorithms/, 2024. Accessed 2026-02-12.

[7] Brian W. Kernighan and Dennis M. Ritchie. *The C Programming Language*. Prentice Hall, 2nd edition, 1988.

[8] Flowchart symbols and notation. https://www.lucidchart.com/blog/flowchart-symbols-and-notation, 2024. Accessed 2026-02-12.

[9] Binary search. https://en.wikipedia.org/wiki/Binary_search, 2024. Accessed 2026-02-12.

[10] Bubble sort. https://en.wikipedia.org/wiki/Bubble_sort, 2024. Accessed 2026-02-12.

[11] Flowchart. https://en.wikipedia.org/wiki/Flowchart, 2024. Accessed 2026-02-12.

[12] Insertion sort. https://en.wikipedia.org/wiki/Insertion_sort, 2024. Accessed 2026-02-12.

[13] Linear search. https://en.wikipedia.org/wiki/Linear_search, 2024. Accessed 2026-02-12.

[14] Outcome-based education. https://en.wikipedia.org/wiki/Outcome-based_education, 2024. Accessed 2026-02-12.

[15] Procedural programming. https://en.wikipedia.org/wiki/Procedural_programming, 2024. Accessed 2026-02-12.

[16] Pseudocode. https://en.wikipedia.org/wiki/Pseudocode, 2024. Accessed 2026-02-12.

[17] Selection sort. https://en.wikipedia.org/wiki/Selection_sort, 2024. Accessed 2026-02-12.

[18] Sorting algorithm. https://en.wikipedia.org/wiki/Sorting_algorithm, 2024. Accessed 2026-02-12.