Jatekom (Jurnal Aplikasi Teknologi dan Komputasi)
Vol. 1, No. 4, Desember 2025, pp. 38~48

Sistem Manajemen Nilai Mahasiswa Berbasis Konsol Menggunakan C++: Pendekatan Prosedural Tipe Data Abstrak

Mohammad Mirza Fahri Sofa[1], Reyhand Ilham Prasetya[2]
[1,2]Program Studi Teknin Informatika, Fakultas Ilmu Komputer, Universitas Dian Nuswantoro, Kota Semarang

Artikel Info

Kata kunci:

C++
Abstract Data Type
Bubble Sort
Pemrograman Prosedural

ABSTRAK

Makalah ini menyajikan perancangan, implementasi, serta analisis teknis dari sistem manajemen nilai mahasiswa berbasis konsol yang dikembangkan menggunakan bahasa pemrograman C++ dengan pendekatan pemrograman prosedural. Pembahasan diawali dengan penjelasan metodologi penelitian yang digunakan, meliputi pendekatan konstruktif, alasan pemilihan bahasa C++ dan paradigma prosedural, strategi perancangan modular secara *top-down*, serta proses pengembangan yang dilakukan secara bertahap. Sistem dirancang dengan memanfaatkan *Abstract Data Type* (ADT) kustom untuk merepresentasikan data akademik mahasiswa sebagai inti struktur data. Implementasi sistem mengintegrasikan berbagai konstruksi dasar pemrograman, antara lain struktur kondisional bersarang, perulangan tunggal dan bersarang, *array* satu dimensi serta *array* dua dimensi implisit, fungsi buatan pengguna, dan prosedur tanpa nilai balik. Untuk keperluan perangkingan mahasiswa berdasarkan Indeks Prestasi Kumulatif (IPK), sistem menggunakan algoritma *bubble sort*. Makalah ini juga menguraikan secara rinci kontribusi masing-masing konsep pemrograman terhadap arsitektur dan fungsionalitas sistem. Selain itu, dibahas pula kompromi desain yang melekat pada pendekatan prosedural, khususnya terkait keterbatasan skalabilitas dan enkapsulasi data. Sebagai penutup, makalah ini mengemukakan beberapa peluang pengembangan lebih lanjut, termasuk penerapan paradigma berorientasi objek, penggunaan penyimpanan data persisten berbasis berkas, serta pemanfaatan algoritma pengurutan yang lebih efisien.

Penulis Korespondensi :

Nama penulis korespondensi,
Program Studi Teknik Informatika
Fakultas Ilmu Komputer
Universitas Dian Nuswantoro, Kota Semarang, 50131
Email: 111202516172@mhs.dinus.ac.id

1. PENDAHULUAN

Pengelolaan data akademik mahasiswa merupakan tugas fundamental dalam setiap institusi pendidikan [1]. Data akademik yang akurat dan terorganisasi dengan baik sangat penting untuk memantau perkembangan mahasiswa, mendukung pengambilan keputusan administratif, menghasilkan transkrip resmi, serta mengevaluasi capaian pembelajaran [2]. Seiring dengan meningkatnya jumlah mahasiswa dan semakin kompleksnya struktur akademik, kebutuhan akan sistem yang andal untuk mencatat, memproses, dan mengambil informasi akademik secara efisien menjadi semakin penting [3].

Dalam konteks pendidikan ilmu komputer, pengembangan sistem pengelolaan data akademik merepresentasikan permasalahan dunia nyata yang bermakna dan dapat diselesaikan secara efektif melalui pemrograman [4]. Sistem semacam ini secara alami menuntut integrasi berbagai konsep inti, termasuk struktur

data, algoritma, alur kendali (*control flow*), dan desain modular [5]. Melalui implementasi sistem manajemen nilai mahasiswa, peserta didik tidak hanya mempelajari sintaks teknis, tetapi juga dilatih dalam dekomposisi masalah secara logis serta praktik pengembangan perangkat lunak yang sistematis [5], [6].

Aplikasi berbasis konsol yang ditulis menggunakan bahasa pemrograman C++ masih memegang peranan penting dalam kurikulum sarjana, khususnya pada mata kuliah pemrograman tingkat dasar dan menengah [7]. Meskipun industri telah banyak mengadopsi antarmuka grafis dan sistem berbasis web, aplikasi konsol tetap bernilai karena kesederhanaan dan transparansinya [8]. Berbeda dengan kerangka kerja tingkat tinggi yang mengabstraksikan banyak detail implementasi, program berbasis konsol menuntut penanganan eksplisit terhadap memori, representasi data, operasi *input*/*output*, serta struktur kendali program [9], [10]. Karakteristik ini menjadikannya sangat efektif sebagai alat pedagogis untuk memperkuat pemahaman prinsip-prinsip dasar pemrograman [11].

Bahasa pemrograman C++ sangat sesuai untuk tujuan tersebut karena mendukung paradigma pemrograman prosedural dan terstruktur, sekaligus menawarkan kinerja tinggi serta kendali yang rinci terhadap sumber daya sistem [12]. Melalui penggunaan struktur (*struct*), *array*, fungsi, dan pernyataan kendali, mahasiswa dapat memodelkan entitas dunia nyata yang kompleks secara jelas dan logis [13]. Selain itu, C++ mendorong organisasi program yang disiplin, yang merupakan aspek penting dalam pengembangan perangkat lunak yang skalabel dan mudah dipelihara [14], [15].

Penelitian ini mendokumentasikan perancangan dan implementasi sebuah sistem manajemen nilai mahasiswa yang dikembangkan sebagai aplikasi C++ berbasis konsol. Sistem ini dirancang untuk mensimulasikan proses-proses akademik yang umum dijumpai di institusi pendidikan tinggi, termasuk pengelolaan identitas mahasiswa, pengolahan nilai dari beberapa mata kuliah, serta evaluasi kinerja akademik. Alih-alih mengandalkan basis data eksternal atau komponen grafis, sistem ini berfokus pada pemrosesan data di memori utama (*in-memory*) guna menekankan kejelasan algoritmik dan desain struktural. Sistem manajemen nilai mahasiswa yang dirancang ini mendukung beberapa operasi inti, yaitu: pemasukan data mahasiswa dengan *input* nilai dari berbagai mata kuliah, pengambilan dan penampilan seluruh data yang tersimpan, pencarian mahasiswa berdasarkan Nomor Induk Mahasiswa (NIM), pembuatan transkrip nilai individu, perhitungan statistik akademik tingkat kelas, serta pemeringkatan mahasiswa berdasarkan Indeks Prestasi Kumulatif (IPK). Setiap fitur diimplementasikan menggunakan konstruksi pemrograman C++ yang terdefinisi dengan baik, seperti *Abstract Data Type* (ADT), *array*, pernyataan kondisional, perulangan iteratif dan bersarang, serta algoritma pengurutan. Dengan mengadopsi filosofi desain prosedural dan modular, sistem ini memastikan bahwa setiap komponen fungsional terpisah secara logis dan dapat dikelola secara independen. Pendekatan ini tidak hanya meningkatkan keterbacaan dan kemudahan pemeliharaan kode, tetapi juga selaras dengan praktik terbaik dalam pendidikan rekayasa perangkat lunak. Sistem yang dihasilkan menunjukkan bagaimana konsep-konsep dasar pemrograman dapat dikombinasikan untuk menghasilkan aplikasi akademik yang fungsional dan bermakna, baik sebagai prototipe praktis maupun sebagai contoh instruksional bagi mahasiswa yang mempelajari pemrograman C++.

Penelitian ini bertujuan untuk: (a) mendeskripsikan desain arsitektur SGMS serta menjelaskan pemetaan data akademik dunia nyata ke dalam struktur data C++; (b) menganalisis peran setiap konstruksi pemrograman dasar seperti percabangan, perulangan, *array*, fungsi, dan ADT dalam sistem; (c) mengevaluasi pemilihan algoritma *bubble sort* untuk pemeringkatan mahasiswa serta membahas kompleksitasnya; dan (d) membandingkan pendekatan prosedural yang digunakan dengan alternatif berbasis pemrograman berorientasi objek. Sistem ini dirancang sebagai aplikasi *single-session* berbasis memori, yang tidak menyimpan data secara permanen antar eksekusi program dan menggunakan *array* berukuran tetap dengan kapasitas maksimum 100 data mahasiswa. Skala penilaian dan ambang batas predikat kelulusan ditentukan secara *hard-coded* berdasarkan konvensi penilaian yang umum digunakan di perguruan tinggi Indonesia. Batasan-batasan ini diterapkan secara sengaja, karena fokus utama penelitian ini adalah pada demonstrasi konsep pemrograman, bukan pada pengembangan perangkat lunak berskala produksi.

2. METODE

Bab ini menjelaskan metodologi penelitian yang digunakan dalam pengembangan sistem manajemen nilai mahasiswa. Metodologi yang diterapkan mencakup pemilihan bahasa pemrograman dan paradigma pengembangan, pendekatan perancangan sistem, strategi pemodelan data, serta tahapan proses pengembangan perangkat lunak yang dilakukan sejak tahap perumusan kebutuhan hingga implementasi akhir. Seluruh keputusan metodologis dalam penelitian ini diarahkan untuk mendukung tujuan utama, yaitu mendemonstrasikan penerapan konsep-konsep dasar pemrograman C++ dalam sebuah sistem akademik yang fungsional dan terstruktur.

2.1 Pendekatan Penelitian

Penelitian ini menggunakan pendekatan konstruktif (*constructive research approach*), di mana kontribusi utama penelitian diwujudkan dalam bentuk artefak perangkat lunak yang berfungsi [16]. Pendekatan ini tidak berfokus pada pengujian hipotesis atau eksperimen empiris, melainkan pada perancangan dan pembangunan sistem yang secara eksplisit merepresentasikan konsep teoritis dalam bentuk implementasi nyata [17]. Metodologi penelitian diarahkan pada proses konstruksi program yang mampu mengintegrasikan berbagai konsep fundamental pemrograman C++, seperti struktur data, fungsi, percabangan, perulangan, dan algoritma pengurutan. Proses penelitian dibagi ke dalam empat tahap berurutan, yaitu: (1) pendefinisian kebutuhan sistem, (2) perancangan sistem, (3) implementasi, dan (4) analisis. Pembagian ini bertujuan untuk memastikan bahwa setiap tahap pengembangan dilakukan secara sistematis dan terdokumentasi dengan baik.

2.2 Bahasa Pemrograman dan Paradigma

Bahasa pemrograman C++ dipilih sebagai bahasa implementasi utama berdasarkan beberapa pertimbangan. Pertama, C++ menyediakan dukungan langsung terhadap pembentukan tipe data buatan pengguna melalui penggunaan *struct*, yang memungkinkan perancangan *Abstract Data Type* (ADT) secara eksplisit [18]. Fitur ini sangat penting dalam memodelkan data akademik mahasiswa secara terstruktur dan terorganisasi. Kedua, C++ mendukung berbagai paradigma pemrograman, termasuk prosedural dan berorientasi objek [19]. Dalam penelitian ini, paradigma pemrograman prosedural dipilih secara sengaja untuk menjaga kejelasan pedagogis dan menekankan penguasaan konsep dasar pemrograman. Pembatasan ini memungkinkan fokus analisis diarahkan pada penggunaan fungsi, prosedur, *array*, dan struktur kendali tanpa kompleksitas tambahan dari konsep objek dan pewarisan.

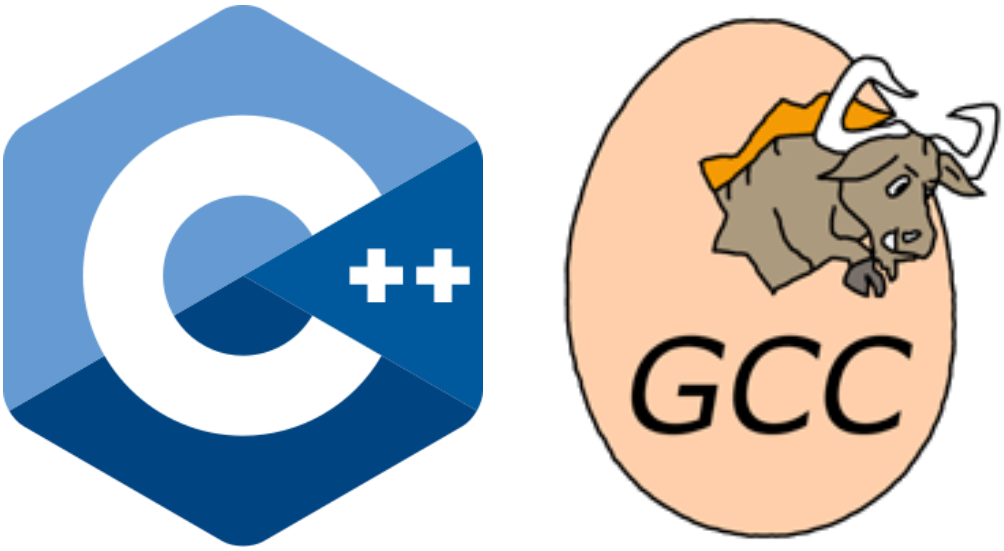

Figure 1. Logo C++ dan GNU

Selain itu, C++ merupakan bahasa pemrograman yang digunakan dalam kurikulum sarjana ilmu komputer, sehingga hasil penelitian ini relevan dan mudah direplikasi dalam konteks pendidikan [20]. Sistem dikompilasi menggunakan kompiler GNU C++ (G++ versi 11.4.0) dengan standar C++17 (-std=c++17) dan dijalankan dalam lingkungan berbasis konsol.

2.3 Metodologi Perancangan Sistem

Perancangan sistem dilakukan menggunakan pendekatan *top-down* dengan dekomposisi modular. Pendekatan *top-down* merupakan metode perancangan perangkat lunak yang dimulai dari perumusan gambaran sistem secara menyeluruh, kemudian secara sistematis diuraikan menjadi komponen-komponen yang lebih kecil dan lebih spesifik [21]. Pada tahap awal, fokus utama diarahkan pada *apa* yang harus dilakukan oleh sistem, bukan langsung pada *bagaimana* implementasi teknisnya [22]. Setelah fungsi utama sistem didefinisikan, setiap fungsi tersebut dipecah menjadi subfungsi hingga mencapai tingkat modul yang cukup sederhana untuk diimplementasikan secara langsung dalam kode program [23]. Dalam konteks sistem ini, dekomposisi *top-down* menghasilkan modul-modul mandiri yang masing-masing menangani satu tanggung

jawab logis, seperti pengolahan data mahasiswa, pencarian data, perhitungan nilai, dan penyajian keluaran. Pendekatan ini membantu mengurangi kompleksitas sistem secara keseluruhan dengan membatasi ruang lingkup setiap modul, sehingga hubungan antarbagian sistem menjadi lebih jelas dan mudah dikendalikan. Dalam implementasinya, setiap modul diwujudkan dalam bentuk fungsi yang mengembalikan nilai atau prosedur tanpa nilai balik (*void*). Struktur ini menciptakan pemisahan yang tegas antara logika navigasi program yang dikelola melalui perulangan menu pada fungsi main() dan logika operasional yang diimplementasikan dalam fungsi-fungsi terpisah. Dengan demikian, pendekatan *top-down* tidak hanya mempermudah proses pengembangan dan pengujian secara bertahap, tetapi juga meningkatkan keterbacaan kode, kemudahan pemeliharaan, serta mendukung prinsip *separation of concerns* dalam rekayasa perangkat lunak.

2.4 Pemodelan Data

Data mahasiswa dimodelkan menggunakan sebuah *struct* dalam bahasa C++ bernama Mahasiswa, yang berperan sebagai *Abstract Data Type* (ADT) inti dalam sistem. *Struct* merupakan struktur data majemuk yang memungkinkan pengelompokan beberapa variabel dengan tipe data yang berbeda ke dalam satu kesatuan logis [24]. Dengan menggunakan *struct*, berbagai atribut yang merepresentasikan satu entitas mahasiswa dapat disatukan dalam satu tipe data terpadu. Dalam sistem ini, struktur Mahasiswa mencakup Nomor Induk Mahasiswa (NIM), nama mahasiswa, sebuah *array* nilai untuk beberapa mata kuliah, nilai IPK hasil perhitungan, serta predikat kelulusan, sehingga merepresentasikan data akademik mahasiswa secara utuh. Untuk menyimpan banyak data mahasiswa, digunakan sebuah *array* berukuran tetap yang terdiri dari elemen bertipe Mahasiswa dengan kapasitas maksimum 100 data. *Array* merupakan struktur data linear yang menyimpan sekumpulan elemen bertipe sama secara berurutan di memori dan dapat diakses menggunakan indeks [25]. Penggunaan array memungkinkan pengelolaan kumpulan data mahasiswa secara terstruktur, mendukung proses iterasi, pencarian, pengurutan, serta perhitungan statistik. Seluruh data disimpan dan diproses di memori utama (*in-memory*), tanpa memanfaatkan berkas eksternal maupun basis data. Pendekatan ini dipilih untuk menjaga fokus penelitian pada penerapan struktur data dasar, manipulasi array, serta implementasi algoritma fundamental, tanpa menambah kompleksitas yang berasal dari mekanisme penyimpanan persisten.

2.5 Proses Pengembangan

Proses pengembangan sistem dilakukan melalui empat tahap utama. Pada tahap pertama, kebutuhan sistem didefinisikan berdasarkan tujuan penelitian yang telah dirumuskan pada Bab Pendahuluan. Tahap ini menghasilkan identifikasi enam fungsi utama sistem, yaitu: pemasukan data mahasiswa, penampilan seluruh data, pencarian berdasarkan NIM, pembuatan transkrip nilai, perhitungan statistik kelas, dan pemeringkatan mahasiswa berdasarkan IPK. Tahap kedua berfokus pada perancangan arsitektur sistem menggunakan pendekatan modular *top-down* sebagaimana dijelaskan pada Subbab 2.3. Hasil dari tahap ini berupa rancangan fungsi dan alur program secara keseluruhan. Tahap ketiga merupakan tahap implementasi, di mana setiap modul direalisasikan dalam bahasa C++ dan diintegrasikan ke dalam aplikasi berbasis menu. Tahap terakhir adalah tahap analisis, di mana sistem yang telah diimplementasikan dievaluasi berdasarkan konstruksi pemrograman yang digunakan dan menjadi dasar pembahasan teknis pada bab selanjutnya.

3. PEMBAHASAN HASIL

Bab ini membahas hasil perancangan dan implementasi sistem manajemen nilai mahasiswa yang dikembangkan berdasarkan metodologi pada bab sebelumnya. Fokus pembahasan diarahkan pada bagaimana konsep-konsep desain sistem, pemodelan data, serta pendekatan modular diwujudkan dalam bentuk arsitektur program C++ yang berjalan pada lingkungan berbasis konsol. Setiap komponen sistem dianalisis tidak hanya dari sisi fungsionalitas, tetapi juga dari perannya dalam mendukung keterbacaan kode, kemudahan pemeliharaan, serta tujuan pedagogis dari pengembangan sistem ini. Untuk memberikan gambaran yang sistematis, pembahasan diawali dengan penjelasan mengenai struktur data utama yang digunakan sebagai *Abstract Data Type*, diikuti dengan pemetaan fungsi-fungsi modular yang membentuk keseluruhan sistem, serta mekanisme alur kontrol berbasis menu. Pendekatan ini memungkinkan analisis dilakukan secara bertahap, mulai dari unit data paling dasar hingga integrasi seluruh modul dalam satu aplikasi yang utuh.

3.1 Perancangan dan Arsitektur Sistem

Sistem manajemen nilai mahasiswa ini dirancang menggunakan pendekatan *top-down* modular, di mana sistem secara keseluruhan diuraikan menjadi sejumlah komponen fungsional yang lebih kecil dan saling terpisah. Pendekatan ini memungkinkan setiap bagian sistem menangani satu tugas logis tertentu, sehingga kompleksitas program dapat dikelola dengan lebih baik. Secara arsitektural, sistem ini menggunakan model kontrol terpusat, di mana fungsi main() berperan sebagai pengendali alur program. Seluruh interaksi pengguna

difasilitasi melalui sebuah menu berbasis teks yang berjalan dalam perulangan, sementara proses-proses operasional seperti pengolahan data, pencarian, perhitungan, dan penampilan hasil didelegasikan ke fungsi-fungsi terpisah. Struktur ini menciptakan pemisahan yang jelas antara logika navigasi program dan logika pemrosesan data. Pendekatan ini tidak hanya meningkatkan keterbacaan kode, tetapi juga mempermudah proses pengujian dan analisis hasil, karena setiap modul dapat dievaluasi secara independen.

3.2 *Abstract Data Type*: *Struct* Mahasiswa

Komponen inti dalam arsitektur SGMS adalah *Abstract Data Type* (ADT) Mahasiswa, yang direalisasikan menggunakan struktur (*struct*) dalam bahasa C++. ADT ini merepresentasikan satu entitas mahasiswa secara utuh, dengan mengelompokkan seluruh atribut akademik yang relevan ke dalam satu unit data. Pada table 1 berikut menunjukkan atribut yang terdapat dalam *struct* Mahasiswa:

Tabel 1. Atribut pada *Abstract Data Type* Mahasiswa

Field	Tipe Data	Deskripsi
nim	string	Nomor Induk Mahasiswa sebagai kunci unik
nama	string	Nama lengkap mahasiswa
nilai[5]	float[]	*Array* nilai untuk lima mata kuliah
ipk	float	Indeks Prestasi Kumulatif hasil perhitungan
predikat	string	Predikat kelulusan berdasarkan IPK

Struct Mahasiswa berfungsi sebagai ADT berbasis nilai (value-based ADT), karena setiap elemen menyimpan data mentah sekaligus data turunan (IPK dan predikat) yang dihitung dari nilai mata kuliah. Penggunaan *array* bertipe Mahasiswa (Mahasiswa dataMahasiswa[100]) memungkinkan sistem menyimpan hingga 100 data mahasiswa dalam memori utama dan berperan sebagai basis data sederhana (*in-memory database*). Pemodelan ini memperlihatkan bagaimana konsep ADT dapat digunakan untuk merepresentasikan objek dunia nyata secara terstruktur tanpa harus menggunakan paradigma berorientasi objek.

3.3 Dekomposisi Fungsi Modular

Untuk mendukung prinsip modularitas, SGMS membagi fungsionalitas sistem ke dalam dua kategori utama, yaitu fungsi yang mengembalikan nilai (*value-returning functions*) dan prosedur tanpa nilai balik (*void procedures*). Pembagian ini membantu memperjelas peran setiap modul dalam sistem. Tabel 2 berikut merangkum fungsi dan prosedur yang digunakan:

Tabel 2. Daftar Fungsi Prosedur pada Sistem Manajemen Nilai Mahasiswa

Fungsi / Prosedur	Tipe	Tanggung Jawab
hitungRataRata()	float	Menghitung rata-rata nilai mata kuliah
tentukanGrade()	char	Mengonversi nilai numerik ke huruf mutu (A–E)
tentukanPredikat()	string	Menentukan predikat kelulusan berdasarkan IPK
cariMahasiswa()	int	Melakukan pencarian linier berdasarkan NIM
*input*DataMahasiswa()	void	Menangani *input* dan validasi data mahasiswa
tampilkanSemuaData()	void	Menampilkan seluruh data mahasiswa
tampilkanTranskrip()	void	Menyusun dan menampilkan transkrip nilai
tampilkanStatistik()	void	Menghitung dan menampilkan statistik kelas
tampilkanRanking()	void	Mengurutkan dan menampilkan peringkat mahasiswa

Fungsi-fungsi yang mengembalikan nilai digunakan untuk proses komputasi murni, seperti perhitungan IPK, penentuan *grade*, dan pencarian data. Sebaliknya, prosedur digunakan untuk operasi yang berorientasi pada interaksi pengguna dan penampilan hasil. Pola ini menunjukkan penerapan prinsip *single responsibility*, di mana setiap fungsi memiliki tujuan yang jelas dan terbatas.

3.4 Alur Kendali Berbasis Menu

Alur kendali utama sistem diimplementasikan dalam fungsi main() menggunakan perulangan do-while. Struktur ini memastikan bahwa menu utama akan terus ditampilkan hingga pengguna memilih opsi

keluar. *Input* pilihan menu diterima melalui cin dan diproses menggunakan struktur switch-case untuk menentukan prosedur yang akan dijalankan. Pendekatan *menu-driven* ini memberikan beberapa keuntungan. Pertama, sistem menjadi mudah digunakan karena setiap fungsi dapat diakses secara eksplisit melalui menu. Kedua, pemisahan antara logika navigasi dan logika operasional membuat kode lebih mudah dikembangkan dan diuji. Setiap prosedur dapat dipanggil secara independen tanpa mempengaruhi alur utama program. Secara keseluruhan, desain alur kendali ini menunjukkan bagaimana struktur kendali dasar dalam C++ dapat digunakan untuk membangun aplikasi interaktif yang terorganisasi dengan baik, meskipun tanpa antarmuka grafis.

3.5 Struktur Kondisional

Struktur kondisional digunakan secara luas dalam sistem untuk mengendalikan alur eksekusi dan memastikan konsistensi data. Kondisional sederhana (*simple if*) digunakan sebagai mekanisme penjagaan (*guard conditions*), misalnya untuk memeriksa apakah jumlah data mahasiswa telah mencapai kapasitas maksimum sebelum proses penambahan data dilakukan (if (jumlahMahasiswa >= 100)), atau untuk menentukan apakah pencarian berdasarkan NIM menghasilkan indeks yang valid.

Tabel 3. Contoh Implementasi Struktur Kondisional

main.cpp

```
if (jumlah == 0) {
        cout << "\nBelum ada data mahasiswa!\n";
        return;
    }

if (nilai >= 85) {
        return 'A';
    } else if (nilai >= 70) {
        return 'B';
    } else if (nilai >= 60) {
        return 'C';
    } else if (nilai >= 50) {
        return 'D';
    } else {
        return 'E';
    }
if (ipk >= 3.5) {
        return "Cum Laude";
    } else if (ipk >= 3.0) {
        return "Sangat Memuaskan";
    } else if (ipk >= 2.5) {
        return "Memuaskan";
    } else if (ipk >= 2.0) {
        return "Cukup";
    } else {
        return "Tidak Lulus";
    }
```

Selain itu, sistem memanfaatkan struktur kondisional bertingkat (*if–else if*) pada fungsi pemetaan nilai dan predikat kelulusan. Fungsi tentukanGrade() membagi rentang nilai numerik kontinu [0–100] ke dalam lima kategori diskret (A sampai E), sedangkan fungsi tentukanPredikat() memetakan nilai IPK ke dalam lima tingkat predikat akademik, mulai dari *Tidak Lulus* hingga *Cum Laude*. Pendekatan ini menunjukkan penerapan klasifikasi berbasis aturan (*rule-based classification*) yang umum digunakan dalam sistem akademik.

3.6 Struktur Perulangan

Sistem ini mengimplementasikan beberapa pola perulangan untuk mendukung pengolahan data yang berulang. Perulangan tunggal (*for loop*) digunakan untuk mengiterasi *array* mata kuliah yang berjumlah tetap, baik dalam proses *input* nilai, perhitungan rata-rata, maupun penampilan data. Perulangan serupa juga digunakan untuk mengakses seluruh data mahasiswa pada proses penampilan dan pencarian.

Tabel 4. Contoh Implementasi Struktur Perulangan

main.cpp

```
for (int i = 0; i < jumlah; i++) {
   for (int j = 0; j < 5; j++) {
      totalPerMatkul[j] += mhs[i].nilai[j];

      if (mhs[i].nilai[j] > tertinggiPerMatkul[j])
         tertinggiPerMatkul[j] = mhs[i].nilai[j];

      if (mhs[i].nilai[j] < terendahPerMatkul[j])
         terendahPerMatkul[j] = mhs[i].nilai[j];
   }
}

while (mhs[jumlah].nilai[i] < 0 || mhs[jumlah].nilai[i] > 100) {
   cout << "Nilai harus 0-100! Masukkan ulang: ";
   cin >> mhs[jumlah].nilai[i];
}
```

Struktur *while loop* digunakan dalam konteks validasi *input*, di mana sistem akan terus meminta masukan dari pengguna hingga nilai yang dimasukkan berada dalam rentang yang diperbolehkan (0–100). Selain itu, perulangan bersarang (*nested loops*) muncul pada dua komponen utama: perhitungan statistik kelas dan proses pengurutan data mahasiswa. Pada fungsi statistik, perulangan luar mengiterasi data mahasiswa, sementara perulangan dalam mengiterasi mata kuliah untuk menghitung nilai rata-rata, tertinggi, dan terendah per mata kuliah. Pola serupa juga digunakan dalam algoritma pengurutan *bubble sort*.

3.7 Pemanfaatan *Array*

Array berperan sebagai struktur penyimpanan utama dalam sistem. *Array* satu dimensi digunakan untuk menyimpan kumpulan data mahasiswa (Mahasiswa dataMahasiswa[100]) serta daftar nama mata kuliah (string namaMataKuliah[5]). Di dalam setiap elemen struktur Mahasiswa, terdapat *array* tetap nilai[5] yang menyimpan nilai tiap mata kuliah.

Tabel 5. Contoh Implementasi Struktur *Array*

main.cpp

```
void inputDataMahasiswa(Mahasiswa mhs[], int &jumlah, string namaMatkul[]) {
   // 1. INPUT OUTPUT
   cout << "\n----- INPUT DATA MAHASISWA -----\n";
   cout << "NIM: ";
   cin >> mhs[jumlah].nim;
   cin.ignore();
   cout << "Nama: ";
   getline(cin, mhs[jumlah].nama);

   // 4. LOOP untuk input nilai 5 mata kuliah
   cout << "\nMasukkan nilai untuk 5 mata kuliah:\n";
   for (int i = 0; i < 5; i++) {
      cout << namaMatkul[i] << ": ";
      cin >> mhs[jumlah].nilai[i];

      // 2. KONDISI - Validasi nilai
      while (mhs[jumlah].nilai[i] < 0 || mhs[jumlah].nilai[i] > 100) {
         cout << "Nilai harus 0-100! Masukkan ulang: ";
         cin >> mhs[jumlah].nilai[i];
      }
   }
```

```
    // Hitung IPK dan predikat
    mhs[jumlah].ipk = hitungRataRata(mhs[jumlah].nilai, 5);
    mhs[jumlah].predikat = tentukanPredikat(mhs[jumlah].ipk);

    cout << "\nData berhasil disimpan!\n";
    jumlah++;
}
```

Kombinasi ini membentuk struktur data dua tingkat (*implicit multi-dimensional structure*), di mana nilai mahasiswa ke-i pada mata kuliah ke-j diakses melalui ekspresi dataMahasiswa[i].nilai[j]. Pola pengindeksan ini menjadi inti dari proses perhitungan statistik kelas dan menunjukkan penerapan konsep *array* bersarang secara praktis dalam konteks pengolahan data akademik.

3.8 Fungsi dan Prosedur

Perancangan sistem secara jelas membedakan antara fungsi yang mengembalikan nilai (*value-returning functions*) dan prosedur tanpa nilai balik (*void procedures*). Tugas-tugas yang bersifat komputasional murni seperti perhitungan rata-rata, penentuan *grade*, penentuan predikat, dan pencarian data diimplementasikan sebagai fungsi yang menerima parameter dan mengembalikan hasil tanpa efek samping.

Tabel 6. Implementasi Fungsi dan Prosedur

main.cpp

```
float hitungRataRata(float nilai[], int jumlahMatkul) {
    float total = 0;
    for (int i = 0; i < jumlahMatkul; i++) {
        total += nilai[i];
    }
    return total / jumlahMatkul;
}
```

Sebaliknya, operasi yang melibatkan interaksi dengan pengguna, seperti pengambilan *input*, penampilan data, dan pencetakan laporan, diimplementasikan sebagai prosedur. Mekanisme *pass-by-reference* digunakan pada prosedur *input*DataMahasiswa() untuk memungkinkan pembaruan langsung terhadap variabel penghitung jumlah data mahasiswa, sehingga aliran data menjadi eksplisit dan mudah ditelusuri.

3.9 Analisis Algoritma Perangkingan (*Bubble sort*)

Prosedur tampilkanRanking() bertugas menyusun data mahasiswa berdasarkan nilai IPK secara menurun untuk menghasilkan peringkat kelas. Algoritma *bubble sort* dipilih sebagai metode pengurutan karena kesederhanaannya dan kemudahan implementasi. Untuk menjaga integritas data asli, sistem terlebih dahulu menyalin seluruh *array* mahasiswa ke dalam *array* sementara sebelum proses pengurutan dilakukan.

Tabel 7. Implementasi Algoritma *Bubble sort*

main.cpp

```
for (int i = 0; i < jumlah - 1; i++) {
    for (int j = 0; j < jumlah - i - 1; j++) {
        if (temp[j].ipk < temp[j + 1].ipk) {
            Mahasiswa swap = temp[j];
            temp[j] = temp[j + 1];
            temp[j + 1] = swap;
        }
    }
}
```

Algoritma *bubble sort* bekerja dengan cara membandingkan pasangan elemen yang bersebelahan dan menukarnya jika urutan tidak sesuai. Proses ini diulang sebanyak $(n - 1)$ iterasi, dengan n menyatakan jumlah mahasiswa. Pada setiap iterasi, elemen dengan nilai IPK terkecil secara bertahap "menggelembung" ke posisi akhir dari bagian *array* yang belum terurut.

Bubble sort memiliki kompleksitas waktu rata-rata dan terburuk sebesar $O(n^2)$, serta kompleksitas terbaik $O(n)$ apabila data telah terurut dan optimasi penghentian dini diterapkan. Kompleksitas ruang algoritma ini adalah $O(1)$ karena pengurutan dilakukan secara *in-place*. Dengan batas maksimum 100 data mahasiswa, kompleksitas kuadratik tersebut tidak menimbulkan dampak kinerja yang signifikan. Selain itu, sifat stabil dari *bubble sort* memastikan bahwa mahasiswa dengan IPK yang sama mempertahankan urutan *input* awal, yang relevan dalam konteks perangkingan akademik.

3.10 Kelebihan Pendekatan Prosedural

Pendekatan prosedural yang digunakan dalam proyek ini memberikan beberapa keunggulan utama. Setiap fungsi memiliki tanggung jawab yang jelas dan terdefinisi dengan baik, sehingga alur program mudah ditelusuri dan dipelihara. Pemisahan antara logika komputasi dan operasi I/O juga memungkinkan pemanfaatan ulang fungsi inti tanpa ketergantungan pada mekanisme tampilan. Selain itu, penggunaan parameter *pass-by-value* dan *pass-by-reference* secara eksplisit membantu mahasiswa memahami aliran data dalam program. Di sisi lain, desain prosedural memiliki keterbatasan inheren. Struktur Mahasiswa tidak menerapkan enkapsulasi, sehingga konsistensi antara nilai, IPK, dan predikat tidak dijamin secara otomatis jika terjadi perubahan data. Penggunaan *array* berukuran tetap membatasi skalabilitas sistem, dan ketiadaan penyimpanan permanen menyebabkan seluruh data hilang ketika program dihentikan. Pendekatan berorientasi objek berpotensi meningkatkan integritas data dengan menjadikan atribut sebagai anggota privat dan menyediakan metode akses terkontrol. Nilai turunan seperti IPK dan predikat dapat dihitung otomatis melalui konstruktor atau *setter*. Namun, untuk ruang lingkup sistem dan tujuan pembelajaran dasar, pendekatan prosedural menawarkan kejelasan yang sebanding dengan kompleksitas struktural yang lebih rendah. Beberapa pengembangan dapat dilakukan untuk meningkatkan fungsionalitas dan ketahanan sistem, antara lain: penambahan penyimpanan permanen berbasis berkas, penggunaan struktur data dinamis seperti std::vector, penerapan algoritma pengurutan yang lebih efisien, refaktorisasi ke paradigma berorientasi objek, serta peningkatan validasi *input* untuk menjamin kualitas data akademik.

4. KESIMPULAN

Proyek ini telah menyajikan analisis yang komprehensif terhadap sebuah sistem manajemen nilai mahasiswa berbasis konsol yang diimplementasikan menggunakan bahasa pemrograman C++ dengan pendekatan pemrograman prosedural. Sistem yang dikembangkan menunjukkan bagaimana konstruksi dasar pemrograman yang meliputi *Abstract Data Type* (ADT), struktur kondisional, perulangan, *array*, fungsi, serta algoritma pengurutan dapat diintegrasikan secara sistematis untuk membangun sebuah aplikasi yang fungsional dan modular. Penggunaan struktur Mahasiswa sebagai ADT berbasis nilai memberikan model yang jelas dalam merepresentasikan entitas dunia nyata yang bersifat kompleks, yaitu data akademik mahasiswa. Selain itu, pemisahan logika program ke dalam fungsi dan prosedur yang memiliki tanggung jawab tunggal terbukti meningkatkan keterbacaan kode, kemudahan pemeliharaan, serta kejelasan alur eksekusi program. Pendekatan ini juga selaras dengan tujuan pedagogis, khususnya dalam konteks pembelajaran pemrograman dasar dan terstruktur. Algoritma *bubble sort* yang digunakan dalam proses perangkingan mahasiswa, meskipun tidak optimal dari sisi kompleksitas waktu, dinilai sesuai dengan skala sistem yang dibatasi hingga 100 data mahasiswa. Keunggulan utama algoritma ini terletak pada kesederhanaan dan transparansi prosesnya, sehingga mudah dipahami dan dianalisis dalam konteks pendidikan. Dengan demikian, pemilihan algoritma tersebut mendukung tujuan pembelajaran tanpa mengorbankan fungsionalitas sistem. Pembahasan mengenai keterbatasan sistem serta usulan pengembangan di masa mendatang menunjukkan adanya jalur yang jelas untuk mentransformasikan prototipe ini menjadi sistem yang lebih tangguh dan mendekati kebutuhan perangkat lunak produksi. Dengan penambahan fitur seperti penyimpanan data persisten, penggunaan struktur data dinamis, serta penerapan paradigma berorientasi objek, sistem ini berpotensi dikembangkan lebih lanjut baik sebagai media pembelajaran lanjutan maupun sebagai dasar implementasi sistem akademik yang lebih komple

REFERENCES

[1] H. Kasim, A. M. Ibrahim, and Z. F. Ahmad, “IMPLEMENTASI TEKNOLOGI BLOCKCHAIN PADA PENGELOLAAN DATA AKADEMIK,” *J. INSTEK Inform. Sains Dan Teknol.*, vol. 10, no. 1, pp. 211–222, May 2025, doi: 10.24252/instek.v10i1.56535.

[2] Arum Masyitoh and Sugeng Pradikto, “Pengaruh Pengelolaan Waktu dan Berorganisasi Terhadap Prestasi Akademik Mahasiswa Pendidikan Ekonomi di Universitas PGRI Wiranegara,” *J. Manaj. Ris. Inov.*, vol. 3, no. 1, pp. 46–58, Jan. 2025, doi: 10.55606/mri.v3i1.3442.

[3] Icha Dwi Rahayu and Kholidiah Kholidiah, "FINTECH PAYMENT, FINANCIAL SELF-EFFICACY DAN KEMAMPUAN AKADEMIK TERHADAP PENGELOLAAN KEUANGAN MAHASISWA SARJANA AKUNTANSI," *J. Ilm. Akunt.*, vol. 3, no. 1, pp. 69–79, Dec. 2025, doi: 10.69714/h96zc008.
[4] A. Nursodiq, R. P. Simanjuntak, A. A. Zaky, D. F. K. Duli, and R. A. Dzikri, "Pengembangan Sistem Prediksi Kelulusan Mahasiswa Tepat Waktu Berdasarkan Data Akademik Menggunakan Metode Jaringan Syaraf Tiruan Backpropagation Berbasis Python," *J. Pendidik. Tambusai*, vol. 10, no. 1, pp. 431–438, Jan. 2026, doi: 10.31004/jptam.v10i1.35885.
[5] Ghofar Taufiq, Yopi Handrianto, and Suharjanti, "Rancang Bangun Sistem Informasi Rekap Data Akademik Mahasiswa dengan Model Extreme Programming," *SATIN - Sains Dan Teknol. Inf.*, vol. 8, no. 1, pp. 42–51, Jun. 2022, doi: 10.33372/stn.v8i1.823.
[6] M. R. A. Fernanda, P. Sokibi, and R. Fahrudin, "SISTEM PREDIKSI KETEPATAN KELULUSAN MAHASISWA BERDASARKAN DATA AKADEMIK DAN NON AKADEMIK MENGGUNAKAN METODE K-MEANS (STUDI KASUS : UNIVERSITAS CATUR INSAN CENDEKIA)," in *Jurnal Digit*, May 2021, p. 89. doi: 10.51920/jd.v11i1.182.
[7] H. Fianto Putra, "Peran, Peluang, Dan Tantangan Penggunaan Kecerdasan Buatan Dalam Pembelajaran Pemrograman Bagi Mahasiswa Sarjana Informatika: Sebuah Tinjauan Literatur," *SUBMIT J. Ilm. Teknol. Infomasi Dan Sains*, vol. 5, no. 2, pp. 23–38, Dec. 2025, doi: 10.36815/submit.v5i2.4528.
[8] M. Jiao *et al.*, "Research on the application of high-speed EMU driver's cab console based on new composite materials," *J. Phys. Conf. Ser.*, vol. 2819, no. 1, p. 012035, Aug. 2024, doi: 10.1088/1742-6596/2819/1/012035.
[9] O. Tsypliak and V. Artemchuk, "Console Application Development for Articles` Highlights Generation Based on Artificial Intelligence Designed Using Autonomous Large Language Model," E. Faure, Y. Tryus, T. Vartiainen, O. Danchenko, M. Bondarenko, C. Bazilo, and G. Zaspa, Eds., in Lecture Notes on Data Engineering and Communications Technologies, vol. 221. Cham: Springer Nature Switzerland, 2024, pp. 53–64. doi: 10.1007/978-3-031-71801-4_5.
[10] K. V. Rozov, A. V. Podsadnikov, and N. A. Chupin, "Story-Based Tasks with Console Visualization in the C# Programming Course as a Way to Activate Creative Independent Work of Students," *Open Educ.*, vol. 29, no. 6, pp. 4–19, Dec. 2025, doi: 10.21686/1818-4243-2025-6-4-19.
[11] A. Kristiyanto *et al.*, "Pelatihan Pemrograman C++ Melalui Tinkercad Guna Meningkatkan Kemampuan Computational Thinking Siswa SMK 11 Kab. Tangerang," *J. Pengabdi. Kpd. Masy. Nusant.*, vol. 5, no. 2, pp. 2776–2782, Jun. 2024, doi: 10.55338/jpkmn.v5i2.3362.
[12] R. Firdaus, D. Rinaldi, H. D. Septama, and S. Maulana, "Pengembangan e-Modul Pemrograman Dasar C++ dalam Upaya Meningkatkan Level of Understanding Siswa," *J. PPS-TP J. Pendidik. Anal. Apl. Teori Dan Has. Penelit.*, vol. 12, no. 2, pp. 17–33, Oct. 2024.
[13] J. J. Siang, L. D. Krisnawati, Y. Lukito, L. Chrisantyo, and R. G. Santoso, "Pelatihan Algoritma Pemrograman Untuk Persiapan OSN Informatika Siswa SMA Kolose de Britto Yogyakarta | Prima Abdika: Jurnal Pengabdian Masyarakat," Nov. 2025, Accessed: Feb. 02, 2026. [Online]. Available: https://e-journal.uniflor.ac.id/index.php/abdika/article/view/6897
[14] W. Purbasari *et al.*, *ALGORITMA PEMROGRAMAN*. CV WIDINA MEDIA UTAMA, 2024. Accessed: Feb. 02, 2026. [Online]. Available: https://repository.penerbitwidina.com/publications/567639/
[15] N. Fahriani, "MENGENAL FUNDAMENTAL PROGRAMMING DENGAN DEVC++," UM Surabaya Publishing, 2022, pp. 171–183. Accessed: Feb. 02, 2026. [Online]. Available: https://repository.um-surabaya.ac.id/id/eprint/9652/
[16] O. Jones, J. Gold, and J. Claxton, "An exposition of the constructive research approach: a tactical treatise for addressing methodological and practical issues in organisational research," *Int. J. Organ. Anal.*, vol. 31, no. 7, pp. 3051–3069, Aug. 2022, doi: 10.1108/IJOA-03-2022-3212.
[17] N. Davis, "A Constructive Approach to Managing Faculty Conflict: An Action Research Study," *Theses Diss.--Educ. Leadersh. Stud.*, Jan. 2021, doi: https://doi.org/10.13023/etd.2021.226.
[18] M. A. Aziz, A. S. Shaarani, S. S. K. Baharin, Z. Othman, and N. H. Hassan, "Building Strong Foundations in C++: Scaffolded Exercises with CodeRunner," in *2024 International Conference on TVET Excellence & Development (ICTeD)*, Dec. 2024, pp. 98–102. doi: 10.1109/ICTeD62334.2024.10844611.
[19] L. de Moura and S. Ullrich, "The Lean 4 Theorem Prover and Programming Language," in *Automated Deduction – CADE 28*, A. Platzer and G. Sutcliffe, Eds., Cham: Springer International Publishing, 2021, pp. 625–635. doi: 10.1007/978-3-030-79876-5_37.
[20] L. Reynolds and K. McDonell, "Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm," in *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing*

Systems, in CHI EA '21. New York, NY, USA: Association for Computing Machinery, May 2021, pp. 1–7. doi: 10.1145/3411763.3451760.

[21] P.-C. Cheng *et al.*, "Intel TDX Demystified: A Top-Down Approach," *ACM Comput Surv*, vol. 56, no. 9, p. 238:1-238:33, Apr. 2024, doi: 10.1145/3652597.

[22] A. Zou *et al.*, "Representation Engineering: A Top-Down Approach to AI Transparency," Mar. 03, 2025, *arXiv*: arXiv:2310.01405. doi: 10.48550/arXiv.2310.01405.

[23] C. Chen *et al.*, "TOPIQ: A Top-Down Approach From Semantics to Distortions for Image Quality Assessment," *IEEE Trans. Image Process.*, vol. 33, pp. 2404–2418, 2024, doi: 10.1109/TIP.2024.3378466.

[24] S. N. Mohanty and P. K. Tripathy, *Data Structure and Algorithms Using C++: A Practical Implementation*. John Wiley & Sons, 2021.

[25] S. Matthai, "How to Write Efficient C++ Simulators Using Arrays of Structures Not Structures of Arrays in OpenCSMP," presented at the ECMOR 2024, European Association of Geoscientists & Engineers, Sep. 2024, pp. 1–21. doi: 10.3997/2214-4609.202437056.