Menelusuri Landasan Kode : Kajian Literatur Terhadap Struktur Data dalam C++

by Agung Yuliyanto Nugroho

Submission date: 06-Sep-2024 08:26AM (UTC+0700)
Submission ID: 2446149157
File name: ALPRO_AGUNG_C_jurnal_blm_publish.docx (1.22M)
Word count: 2903
Character count: 19060

Menelusuri Landasan Kode : Kajian Literatur Terhadap Struktur Data dalam C++

Agung Yuliyanto Nugroho1 Nur Hamid Sutanto2
Prodi. Informatika Universitas Cendekia Mitra Indonesia
Prodi. Sistem Informasi Fakultas Ilmu Komputer Universitas Amikom Yogyakarta
E-mail: agungyuliyanto@unicimi.ac.id nurhamid@amikom.ac.id

ABSTRAK

Penelitian ini bertujuan untuk menelusuri landasan kode dalam bahasa pemrograman C++ dengan fokus pada struktur data. Struktur data merupakan komponen fundamental dalam pengembangan perangkat lunak yang efisien dan efektif. Kajian literatur ini mengeksplorasi berbagai jenis struktur data yang diimplementasikan dalam C++, termasuk array, linked list, stack, queue, hash table, dan tree. Analisis dilakukan terhadap prinsip dasar dan penerapan struktur data tersebut dalam konteks C++, dengan memperhatikan kekuatan dan kelemahan masing-masing struktur. Selain itu, penelitian ini juga membahas evolusi dan perbandingan implementasi struktur data dari perspektif kinerja dan kompleksitas algoritma. Hasil dari kajian ini memberikan wawasan mendalam tentang bagaimana struktur data dapat mempengaruhi desain dan efisiensi aplikasi C++, serta memberikan panduan bagi pengembang perangkat lunak dalam memilih struktur data yang paling sesuai untuk kebutuhan spesifik mereka. Kesimpulan dari penelitian ini menyoroti pentingnya pemahaman mendalam tentang struktur data dalam pengembangan perangkat lunak dan kontribusinya terhadap keberhasilan proyek perangkat lunak.

Kata Kunci : Struktur Data; C++; Kinerja Struktur Data

ABSTRACT

This study aims to explore the foundation of code in the C++ programming language with a focus on data structures. Data structures are a fundamental component in efficient and effective software development. This literature review explores various types of data structures implemented in C++, including arrays, linked lists, stacks, queues, hash tables, and trees. An analysis is conducted on the basic principles and implementation of these data structures in the context of C++, taking into account the strengths and weaknesses of each structure. In addition, this study also discusses the evolution and comparison of data structure implementations from the perspective of performance and algorithm complexity. The results of this study provide in-depth insights into how data structures can affect the design and efficiency of C++ applications, as well as provide guidance for software developers in choosing the most appropriate data structures for their specific needs. The conclusions of this study highlight the importance of a deep understanding of data structures in software development and their contribution to the success of software projects.

Keyword : Data Structure; C++; Data Structure Performance

PENDAHULUAN

Dalam dunia pengembangan perangkat lunak, struktur data memegang peranan penting dalam menentukan efisiensi dan efektivitas sebuah aplikasi. Struktur data adalah cara untuk mengorganisasi dan menyimpan data agar dapat diakses dan dimanipulasi secara efisien. Dalam bahasa pemrograman C++, struktur data sering digunakan untuk mengimplementasikan algoritma yang kompleks, serta mendukung berbagai operasi dasar yang diperlukan dalam pemrograman.

C++ adalah bahasa pemrograman yang dikenal karena kemampuannya dalam menyediakan kontrol yang mendalam terhadap memori dan performa. Bahasa ini menawarkan berbagai jenis struktur data yang dapat diimplementasikan oleh programmer untuk memenuhi kebutuhan spesifik dari aplikasi mereka. Struktur data seperti array, linked list, stack, queue, hash table, dan tree masing-masing memiliki karakteristik unik yang membuatnya lebih cocok untuk jenis masalah tertentu. Pemilihan struktur data yang tepat sangat penting dalam merancang sistem yang efisien dan responsif.

Kajian literatur ini bertujuan untuk mengeksplorasi berbagai struktur data yang tersedia dalam C++ serta memahami prinsip-prinsip dasar dan aplikasi praktis dari masing-masing struktur. Fokus utama dari penelitian ini adalah untuk meneliti bagaimana struktur data tersebut diimplementasikan, kelebihan dan kekurangan mereka dalam konteks C++, serta dampaknya terhadap kinerja dan kompleksitas algoritma. Melalui pemahaman yang mendalam tentang struktur data ini, pengembang dapat membuat keputusan yang lebih baik dalam merancang dan mengoptimalkan aplikasi mereka.

Dengan menelaah berbagai literatur yang ada, kajian ini akan memberikan panduan yang komprehensif tentang struktur data dalam C++, serta kontribusinya terhadap pengembangan perangkat lunak yang efektif. Hasil dari kajian ini diharapkan dapat menjadi referensi berharga bagi para pengembang dan peneliti dalam bidang pemrograman dan rekayasa perangkat lunak.

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

// Definisi struktur
struct Person {
    string name;
    int age;
    float height;
};

int main() {
    // Deklarasi dan inisialisasi variabel struktur
    Person person1;
    person1.name = "Alice";
    person1.age = 30;
    person1.height = 5.5;

    // Mengakses anggota struktur
    cout << "Name: " << person1.name << endl;
    cout << "Age: " << person1.age << endl;
    cout << "Height: " << person1.height << endl;

    return 0;
}
```

Gambar 1. Struktur dalam C++:

A. Komponen Struktur

- Anggota (Members): Variabel yang didefinisikan dalam struktur. Dalam contoh di atas, name, age, dan height adalah anggota dari struktur Person.
- Instance: Variabel yang dideklarasikan dengan tipe struktur. Di atas, person1 adalah instance dari struktur Person.

B. Constructor dan Destructor dalam Struktur

Seperti dalam kelas, Anda juga dapat mendefinisikan konstruktor dan destruktor dalam struktur untuk menginisialisasi dan membersihkan sumber daya.

```cpp
struct Rectangle {
    int width, height;

    // Konstruktor
    Rectangle(int w, int h) : width(w), height(h) {}

    // Fungsi anggota
    int area() {
        return width * height;
    }
};

int main() {
    Rectangle rect(10, 5);
    cout << "Area: " << rect.area() << endl;

    return 0;
}
```

Gambar 2. Constructor dan Destructor dalam Struktur

C. Struktur vs. Kelas

- Default Access Control: Dalam struktur, anggota memiliki akses publik (public) secara default, sementara dalam kelas (class), akses default adalah privat (private).
- Penggunaan: Struktur umumnya digunakan untuk data yang bersifat sederhana atau data yang tidak memerlukan enkapsulasi yang ketat. Kelas sering digunakan untuk mendefinisikan tipe data yang lebih kompleks dengan kontrol akses dan fungsi anggota.

D. Anggota Struktur

Anggota struktur bisa berupa variabel dari berbagai tipe data. Kamu dapat memiliki berbagai jenis data seperti int, float, string, atau bahkan struktur lain di dalam struktur.

```cpp
struct Rectangle {
    float length;
    float width;
};
```

Gambar 4 Anggota Struktur

E. Akses Anggota Struktur

Anggota struktur dapat diakses menggunakan operator titik (.) pada variabel struktur.

```cpp
Rectangle rect;
rect.length = 10.5;
rect.width = 4.2;
```

Gambar 5Akses Anggota Struktur

F. Struktur dalam Fungsi

Struktur dapat digunakan sebagai parameter fungsi atau nilai kembali dari fungsi.

```cpp
void printRectangle(Rectangle r) {
    cout << "Length: " << r.length << endl;
    cout << "Width: " << r.width << endl;
}
```

Gambar 6 Struktur dalam Fungsi

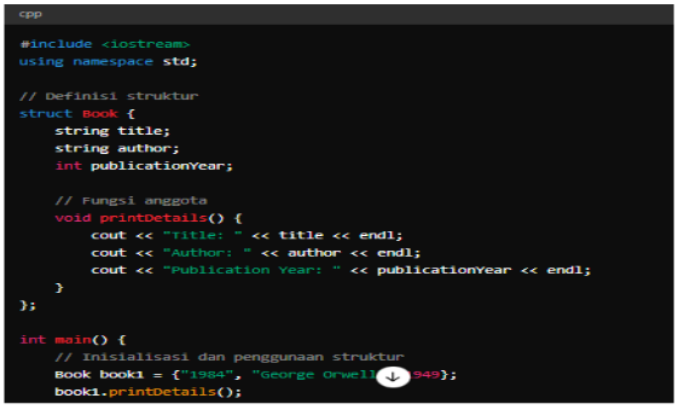
G. Menggunakan Struktur dalam Program

Struktur dapat digunakan untuk:

- Menyimpan Data Kompleks: Memudahkan pengelolaan dan pemrosesan data yang memiliki beberapa atribut.

Mendefinisikan Tipe Data Kustom: Membantu menciptakan tipe data yang lebih bermakna dan sesuai dengan kebutuhan program.

- Fungsi Anggota: Struktur dapat memiliki fungsi anggota untuk melakukan operasi pada data mereka, mirip dengan metode dalam kelas.



Gambar 7 Menggunakan Struktur dalam Program

H. Contoh Penggunaan Struktur

Berikut adalah contoh sederhana dari penggunaan struktur untuk menyimpan data tentang buku:

I. Manfaat Menggunakan Struktur

- **Mengelompokan Data:** Struktur memungkinkan pengelompokan data yang saling terkait, memudahkan manajemen dan pemrosesan data.
- **Organisasi Kode:** Struktur membantu dalam mengorganisasi kode dengan cara yang lebih bersih dan terstruktur.
- **Kemudahan Akses:** Anggota struktur dapat diakses dengan mudah menggunakan operator titik, membuat kode lebih intuitif dan mudah dibaca.

J. Menggunakan Struktur dalam Konteks Program

Struktur sering digunakan dalam berbagai konteks pemrograman, seperti:

- **Menyimpan Data Pengguna:** Seperti informasi pribadi dalam aplikasi.
- **Mendefinisikan Entitas dalam Game:** Seperti karakter atau objek.
- **Pengelolaan Data Dalam Basis Data:** Mengelompokkan data dalam tabel atau rekaman.

Dengan memahami dan menggunakan struktur dalam C++, Anda dapat menyusun data dengan cara yang lebih logis dan terstruktur, yang membuat program lebih mudah dipelihara dan dikembangkan. Struktur adalah salah satu fondasi dari pemrograman berorientasi objek dan konsep-konsep terkait dalam C++.

KAJIAN PUSTAKA

Kajian pustaka ini bertujuan untuk memberikan pemahaman menyeluruh tentang struktur data dalam bahasa pemrograman C++, dengan fokus pada teori dasar, implementasi, serta aplikasi praktis dari berbagai struktur data yang umum digunakan. Beberapa aspek penting dari kajian ini meliputi:

1. Teori Dasar Struktur Data

Struktur data adalah cara sistematis untuk mengorganisir, menyimpan, dan mengelola data. Teori dasar struktur data mencakup definisi dan karakteristik utama dari berbagai jenis struktur data, seperti array, linked list, stack, queue, hash table, dan tree (Cormen et al., 2009; Knuth, 1998). Buku-buku teks seperti *Introduction to Algorithms* oleh Cormen et al. (2009) dan *The Art of Computer Programming* oleh Knuth (1998) sering digunakan sebagai referensi untuk memahami prinsip dasar ini.

2. Array

Array adalah struktur data yang paling sederhana dan sering digunakan. Array memungkinkan penyimpanan elemen dengan tipe data yang sama dalam urutan kontigu di memori. Buku-buku seperti *Data Structures and Algorithm Analysis in C++* oleh Weiss (2014) membahas implementasi dan penggunaan array dalam konteks C++. Array menawarkan akses yang cepat dan efisien, tetapi memiliki batasan dalam hal ukuran tetap dan keterbatasan dalam operasi penyisipan dan penghapusan.

3. Linked List

Linked list adalah struktur data yang memungkinkan penyimpanan elemen secara dinamis dengan setiap elemen yang disebut node berisi referensi (link) ke elemen berikutnya. Implementasi dan analisis linked list dibahas dalam buku seperti *Data Structures and Algorithmic Thinking* oleh Narasimha Karumanchi (2011). Linked list menawarkan fleksibilitas dalam hal ukuran dinamis tetapi dengan overhead memori tambahan untuk referensi dan akses yang lebih lambat dibandingkan array.

4. Stack dan Queue

Stack dan queue adalah struktur data berbasis konsep LIFO (Last In First Out) dan FIFO (First In First Out), masing-masing. Buku seperti *Data Structures and Algorithms in C++* oleh Adam Drozdek (2012) memberikan penjelasan mendalam tentang implementasi stack dan queue serta aplikasi mereka dalam berbagai algoritma dan pemrograman. Stack sering digunakan dalam rekursi dan pemrograman backtracking, sementara queue digunakan dalam antrian tugas dan pemrograman berbasis waktu nyata.

5. Hash Table

Hash table adalah struktur data yang menawarkan pencarian, penyisipan, dan penghapusan data dengan waktu konstan rata-rata. *Algorithms* oleh Sedgewick dan Wayne (2011) menjelaskan bagaimana hash table bekerja, termasuk teknik hashing dan penanganan tabrakan. Hash table sangat berguna dalam aplikasi yang memerlukan operasi pencarian yang cepat, tetapi memerlukan pemahaman tentang fungsi hash dan manajemen tabrakan.

6. Tree

Tree adalah struktur data hierarkis yang terdiri dari node dengan hubungan parent-child. Buku *Introduction to Algorithms* oleh Cormen et al. (2009) dan *The Algorithm Design Manual* oleh Skiena (2009) membahas berbagai jenis tree seperti binary tree, AVL tree, dan B-tree, serta aplikasinya dalam penyimpanan data yang terurut dan pencarian efisien.

7. Perbandingan dan Evaluasi

Perbandingan antara struktur data dilakukan berdasarkan kriteria seperti kompleksitas waktu untuk operasi dasar (insertion, deletion, search) dan penggunaan memori. Artikel-artikel penelitian seperti *Comparative Evaluation of Data Structures for Search Problems* oleh Meyer et al. (2015) sering membahas evaluasi empiris dan teoretis dari struktur data dalam berbagai konteks aplikasi.

8. Aplikasi Praktis

Implementasi struktur data dalam C++ sering kali diterapkan dalam konteks pengembangan perangkat lunak nyata. Buku seperti *Effective STL* oleh Meyers (2001) memberikan panduan praktis tentang penggunaan struktur data standar di C++ STL (Standard Template Library), yang memungkinkan programmer untuk memanfaatkan struktur data yang telah dioptimalkan dan diuji secara luas.

Struktur dan Template

1. **Pengenalan Template**

Template di C++ adalah fitur yang memungkinkan kamu untuk menulis kode yang dapat bekerja dengan berbagai tipe data. Template dapat digunakan untuk fungsi, kelas, dan struktur.

2. **Template Kelas**

Memungkinkan kamu untuk mendefinisikan kelas yang bekerja dengan tipe data generik.

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

template <typename T>
struct Pair {
    T first;
    T second;

    Pair(T f, T s) : first(f), second(s) {}

    void display() {
        cout << "First: " << first << ", Second: " << second << endl;
    }
};

int main() {
    Pair<int> intPair(1, 2);
    intPair.display();

    Pair<string> stringPair("Hello", "World");
    stringPair.display();

    return 0;
}
```

Gambar 7 Struktur Template Sederhana

Pada contoh di atas, struktur Pair adalah struktur template yang dapat menyimpan dua nilai dari tipe data yang sama. Struktur ini dapat diinstansiasi dengan berbagai tipe data, seperti int dan string.

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

template <typename T1, typename T2>
struct Dual {
    T1 first;
    T2 second;

    Dual(T1 f, T2 s) : first(f), second(s) {}

    void display() {
        cout << "First: " << first << ", Second: " << second << endl;
    }
};

int main() {
    Dual<int, double> myDual(1, 3.14);
    myDual.display();

    Dual<string, int> anotherDual("Age", 30);
    anotherDual.display();

    return 0;
}
```

Gambar 8 Struktur Template dengan Tipe yang Berbeda

Pada contoh ini, struktur Dual menggunakan dua template parameter (T1 dan T2) untuk menyimpan nilai dari tipe data yang berbeda.

3. **Fungsi Anggota dalam Struktur Template**

   Seperti kelas, struktur template juga dapat memiliki fungsi anggota.

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

template <typename T>
struct Container {
    T value;

    Container(T v) : value(v) {}

    void setValue(T v) {
        value = v;
    }

    T getValue() {
        return value;
    }

    void display() {
        cout << "Value: " << value << endl;
    }
};

int main() {
    Container<int> intContainer(5);
    intContainer.display();
    intContainer.setValue(10);
    intContainer.display();

    Container<string> stringContainer("Hello");
    stringContainer.display();
    stringContainer.setValue("World");
    stringContainer.display();

    return 0;
}
```

Gambar 9 Fungsi Anggota dalam Struktur Template

Pada contoh ini, `Container` adalah struktur template yang memiliki fungsi anggota untuk mengatur dan mendapatkan nilai, serta menampilkan nilai tersebut.

Struktur template sangat berguna ketika kamu ingin membuat struktur yang dapat bekerja dengan berbagai tipe data tanpa harus menulis ulang kode untuk setiap tipe data. Ini meningkatkan keterbacaan kode dan mengurangi duplikasi.

METODE PENELITIAN

Metode penelitian ini dirancang untuk melakukan kajian literatur yang mendalam mengenai struktur data dalam bahasa pemrograman C++, dengan fokus pada teori, implementasi, dan aplikasi praktis. Metode penelitian yang digunakan dalam kajian ini mencakup langkah-langkah berikut:

1. Penentuan Topik dan Ruang Lingkup

Langkah awal adalah menentukan topik penelitian yang spesifik, yaitu struktur data dalam C++. Ruang lingkup kajian ini mencakup jenis-jenis struktur data yang umum digunakan dalam C++ seperti array, linked list, stack, queue, hash table, dan tree. Fokus utama adalah pada teori dasar, implementasi dalam C++, dan evaluasi aplikasi praktis dari masing-masing struktur data.

2. Pengumpulan Literatur

Literatur yang relevan dikumpulkan dari berbagai sumber seperti buku teks, jurnal akademik, artikel konferensi, dan sumber online terpercaya. Sumber-sumber ini mencakup:
- Buku teks tentang struktur data dan algoritma, seperti *Introduction to Algorithms* oleh Cormen et al. (2009) dan *Data Structures and Algorithm Analysis in C++* oleh Weiss (2014).
- Artikel ilmiah dan studi kasus yang membahas implementasi dan aplikasi struktur data dalam C++.
- Dokumentasi dan panduan resmi dari C++ Standard Template Library (STL).

3. Kriteria Seleksi Literatur

Literatur yang dipilih dievaluasi berdasarkan kriteria berikut:
- Relevansi terhadap topik struktur data dalam C++.
- Kualitas dan kredibilitas sumber (misalnya, publikasi oleh penerbit akademik terkemuka atau kontribusi oleh pakar di bidangnya).
- Kesegaran informasi (terutama untuk topik yang mengalami perkembangan cepat seperti teknologi pemrograman).

4. Analisis dan Sintesis

Setelah mengumpulkan literatur, langkah berikutnya adalah menganalisis dan mensintesis informasi yang ditemukan. Proses ini melibatkan:
- Ringkasan teori dan prinsip dasar dari setiap jenis struktur data yang dibahas.

- Evaluasi implementasi struktur data dalam C++, termasuk kekuatan dan kelemahan masing-masing struktur.
- Perbandingan antara struktur data berdasarkan kriteria seperti kompleksitas waktu dan penggunaan memori.
- Analisis aplikasi praktis dan relevansi struktur data dalam konteks pengembangan perangkat lunak menggunakan C++.

5. Pengorganisasian Temuan

Temuan dari analisis dan sintesis informasi diorganisasi dalam bentuk:
- Tabel perbandingan untuk struktur data, menampilkan kelebihan, kekurangan, dan aplikasi tipikal.
- Diagram atau ilustrasi yang menunjukkan struktur dan operasi dasar dari setiap jenis struktur data.
- Ringkasan temuan kunci dari literatur yang menggarisbawahi tren dan best practices dalam penggunaan struktur data di C++.

6. Penulisan Laporan

Laporan akhir dikembangkan untuk menyajikan hasil kajian literatur dengan struktur yang jelas, termasuk:
- Pendahuluan yang menjelaskan tujuan dan ruang lingkup kajian.
- Kajian pustaka yang merangkum literatur yang relevan.
- Metode penelitian yang menjelaskan proses pengumpulan dan analisis data.
- Diskusi hasil yang membahas temuan utama dan implikasinya.
- Kesimpulan yang menyimpulkan temuan dan memberikan rekomendasi.

7. Verifikasi dan Validasi

Sebelum publikasi, laporan diperiksa dan divalidasi untuk memastikan akurasi dan konsistensi informasi. Ulasan oleh rekan atau ahli di bidang ini mungkin dilakukan untuk mendapatkan umpan balik tambahan dan memastikan kualitas laporan.

8. Publikasi dan Penyebaran

Laporan akhir dipublikasikan dalam bentuk artikel atau laporan penelitian yang dapat diakses oleh komunitas akademik dan profesional. Penyebaran dilakukan melalui platform publikasi akademik atau forum terkait untuk memastikan temuan dapat digunakan dan dinilai oleh pihak yang berkepentingan.

SIMPULAN

Kajian literatur ini telah memberikan wawasan mendalam tentang struktur data dalam bahasa pemrograman C++, serta implementasi dan aplikasi praktisnya. Berdasarkan hasil analisis dan sintesis informasi dari berbagai sumber literatur, beberapa kesimpulan utama dapat ditarik sebagai berikut:

1. Pentingnya Struktur Data dalam C++

Struktur data merupakan komponen fundamental dalam pengembangan perangkat lunak yang efisien. Dalam C++, berbagai jenis struktur data, seperti array, linked list, stack, queue, hash table, dan tree, masing-masing memiliki karakteristik unik yang mempengaruhi bagaimana data disimpan, diakses, dan dimanipulasi. Pemahaman yang mendalam tentang struktur data ini sangat penting untuk merancang aplikasi yang efektif dan efisien.

2. Kelebihan dan Kekurangan Struktur Data

- Array: Menawarkan akses cepat dan sederhana dengan waktu akses konstan, tetapi memiliki keterbatasan dalam hal ukuran tetap dan fleksibilitas penyisipan/penghapusan.

- Linked List: Memungkinkan ukuran dinamis dan operasi penyisipan/penghapusan yang efisien, tetapi dengan overhead memori tambahan dan akses yang lebih lambat dibandingkan array.

- Stack dan Queue: Masing-masing menggunakan prinsip LIFO dan FIFO, menyediakan cara yang efisien untuk mengelola data dalam konteks tertentu, seperti rekursi (stack) dan antrian (queue).

- Hash Table: Menawarkan pencarian dan manipulasi data yang cepat dengan kompleksitas waktu rata-rata konstan, namun memerlukan manajemen tabrakan dan pemilihan fungsi hash yang baik.

- Tree: Struktur hierarkis yang efektif untuk penyimpanan data terurut dan operasi pencarian yang efisien, dengan berbagai jenis tree (seperti binary tree, AVL tree, dan B-tree) menawarkan berbagai keuntungan sesuai kebutuhan aplikasi.

3. Implementasi dalam C++

C++ menyediakan dukungan untuk struktur data melalui Standard Template Library (STL), yang menawarkan implementasi yang sudah teruji dan dioptimalkan dari berbagai struktur data. Pemanfaatan STL memudahkan pengembang untuk mengimplementasikan struktur data tanpa harus membangun dari nol, dan memastikan kompatibilitas serta efisiensi dalam aplikasi.

4. Dampak terhadap Kinerja Aplikasi

Pemilihan struktur data yang tepat memiliki dampak signifikan terhadap kinerja aplikasi. Struktur data yang sesuai dapat meningkatkan efisiensi operasi dasar seperti pencarian, penyisipan, dan penghapusan, serta mengurangi kompleksitas algoritma secara keseluruhan. Oleh karena itu, pemilihan struktur data harus didasarkan pada karakteristik data dan kebutuhan spesifik aplikasi.

5. Rekomendasi untuk Pengembang

- Analisis Kebutuhan: Pengembang harus melakukan analisis mendalam tentang kebutuhan aplikasi sebelum memilih struktur data. Faktor-faktor seperti ukuran data, frekuensi operasi, dan kompleksitas algoritma harus dipertimbangkan.

- Pemanfaatan STL: Memanfaatkan fitur STL dalam C++ untuk struktur data yang umum dapat mempercepat pengembangan dan meningkatkan kualitas kode.

- Evaluasi Kinerja: Selalu evaluasi kinerja struktur data dalam konteks aplikasi nyata dan lakukan pengujian untuk memastikan bahwa pilihan struktur data memenuhi kriteria efisiensi yang diinginkan.

DAFTAR PUSTAKA

BUKU

Stroustrup, B. (2013). The C++ programming language (4th ed.). Addison-Wesley.
Schildt, H. (2019). C++: The complete reference (5th ed.). McGraw-Hill Education.
Stroustrup, B. (2000). The C++ programming language (3rd ed.). Addison-Wesley.

Meyer, B. (2006). Effective C++: 55 specific ways to improve your programs and designs (3rd ed.). Addison-Wesley.

JURNAL

Sahni, S. (2005). Data Structures, Algorithms, and Applications in C++. CRC Press.

Goodrich, M. T., Tamassia, R., & Goldwasser, M. H. (2013). Data Structures and Algorithms in C++. Wiley.

Deitel, P. J., & Deitel, H. M. (2012). C++ How to Program. Pearson.

Stroustrup, B. (2013). The C++ Programming Language. Addison-Wesley.

Knuth, D. E. (1997). The Art of Computer Programming, Volume 1: Fundamental Algorithms. Addison-Wesley.

Bjarne Stroustrup (2020). Programming: Principles and Practice Using C++. Addison-Wesley.

McConnell, S. (2004). Code Complete: A Practical Handbook of Software Construction. Microsoft Press.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms. MIT Press.

GeeksforGeeks (2024). Data Structures and Algorithms in C++. Diakses dari: https://www.geeksforgeeks.org/data-structures/

Stack Overflow (2024). Questions and Discussions on Data Structures in C++. Diakses dari: https://stackoverflow.com/questions/tagged/c++

cppreference.com. (n.d.). C++ reference. Retrieved August 26, 2024, from https://en.cppreference.com/w/

cplusplus.com. (n.d.). C++ reference. Retrieved August 26, 2024, from http://www.cplusplus.com/

GeeksforGeeks. (n.d.). Structures in C++. Retrieved August 26, 2024, from https://www.geeksforgeeks.org/structures-in-c/

Udemy. (n.d.). Beginning C++ programming - From beginner to beyond. Retrieved August 26, 2024, from https://www.udemy.com/course/beginning-c-plus-plus-programming/

Coursera. (n.d.). C++ for C programmers. Retrieved August 26, 2024, from https://www.coursera.org/learn/c-plus-plus-a

ISO/IEC. (n.d.). ISO/IEC C++ standard. Retrieved August 26, 2024, from https://www.iso.org/standard/68564.html

Stroustrup, B. (2013). A tour of C++. Addison-Wesley.

Menelusuri Landasan Kode : Kajian Literatur Terhadap Struktur Data dalam C++

ORIGINALITY REPORT

8%	8%	1%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	apbsrilanka.org Internet Source	3%
2	www.307bwassoc.org Internet Source	1%
3	Submitted to Universitas Sultan Ageng Tirtayasa Student Paper	1%
4	sefidvash.net Internet Source	1%
5	medlib.korea.ac.kr Internet Source	1%
6	Hartatik Hartatik, Arifa Satria Dwi Cahya. "Clusterisasi Kerusakan Gempa Bumi di Pulau Jawa Menggunakan SOM", Jurnal Ilmiah Intech : Information Technology Journal of UMUS, 2020 Publication	1%
7	repository.ar-raniry.ac.id Internet Source	<1%

8	download.garuda.ristekdikti.go.id Internet Source	<1%
9	journal.um-surabaya.ac.id Internet Source	<1%
10	www.slideshare.net Internet Source	<1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		

Menelusuri Landasan Kode : Kajian Literatur Terhadap Struktur Data dalam C++

GRADEMARK REPORT

FINAL GRADE

/0

GENERAL COMMENTS

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16