FUTURE SCIENCE

ALGORITMA DAN PEMROGRAMAN DENGAN C++

Editor : Tundo, M.Kom.

Penulis :

Norbertus Tri Suswanto Saptadi | Ma'shum Abdul Jabbar
Muhammad Dedi Irawan | Erna Hudianti Pujiarini
Mohammad Badrul | Susi Erlinda | Yusuf Ramadhan Nasution
Johan Suryo Prayogo | Cosmas Haryawan | Hidayatus Sibyan
M. Agus Triawan | Tundo | Agung Yuliyanto Nugroho

Bunga Rampai

ALGORITMA DAN PEMROGRAMAN DENGAN C++

UU No 28 tahun 2014 tentang Hak Cipta

Fungsi dan sifat hak cipta Pasal 4

Hak Cipta sebagaimana dimaksud dalam Pasal 3 huruf a merupakan hak eksklusif yang terdiri atas hak moral dan hak ekonomi.

Pembatasan Pelindungan Pasal 26

Ketentuan sebagaimana dimaksud dalam Pasal 23, Pasal 24, dan Pasal 25 tidak berlaku terhadap:

i. Penggunaan kutipan singkat Ciptaan dan/atau produk Hak Terkait untuk pelaporan peristiwa aktual yang ditujukan hanya untuk keperluan penyediaan informasi aktual;
ii. Penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk kepentingan penelitian ilmu pengetahuan;
iii. Penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk keperluan pengajaran, kecuali pertunjukan dan Fonogram yang telah dilakukan Pengumuman sebagai bahan ajar; dan
iv. Penggunaan untuk kepentingan pendidikan dan pengembangan ilmu pengetahuan yang memungkinkan suatu Ciptaan dan/atau produk Hak Terkait dapat digunakan tanpa izin Pelaku Pertunjukan, Produser Fonogram, atau Lembaga Penyiaran.

Sanksi Pelanggaran Pasal 113

1. Setiap Orang yang dengan tanpa hak melakukan pelanggaran hak ekonomi sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf i untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 1 (satu) tahun dan/atau pidana denda paling banyak Rp100.000.000 (seratus juta rupiah).
2. Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf c, huruf d, huruf f, dan/atau huruf h untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan/atau pidana denda paling banyak Rp500.000.000,00 (lima ratus juta rupiah).

ALGORITMA DAN PEMROGRAMAN DENGAN C++

Penulis:

Norbertus Tri Suswanto Saptadi
Ma'shum Abdul Jabbar
Muhammad Dedi Irawan
Erna Hudianti Pujiarini
Mohammad Badrul
Susi Erlinda
Yusuf Ramadhan Nasution
Johan Suryo Prayogo
Cosmas Haryawan
Hidayatus Sibyan
M. Agus Triawan
Tundo
Agung Yuliyanto Nugroho

Editor:

Tundo, M.Kom.

FUTURE SCIENCE

ALGORITMA DAN PEMROGRAMAN DENGAN C++

Penulis:

Norbertus Tri Suswanto Saptadi
Ma'shum Abdul Jabbar
Muhammad Dedi Irawan
Erna Hudianti Pujiarini
Mohammad Badrul
Susi Erlinda
Yusuf Ramadhan Nasution
Johan Suryo Prayogo
Cosmas Haryawan
Hidayatus Sibyan
M. Agus Triawan
Tundo
Agung Yuliyanto Nugroho

Editor: **Tundo, M.Kom.**

Desain Cover: **Nada Kurnia, S.I.Kom.**

Tata Letak: **Samuel, S.Kom.**

Ukuran: **A5 Unesco (15,5 x 23 cm)**

Halaman: **xiv, 211**

e-ISBN: **978-634-7037-98-5**

Terbit Pada: **April 2025**

PENERBIT FUTURE SCIENCE
(CV. FUTURE SCIENCE)
Anggota IKAPI (348/JTI/2022)
Jl. Terusan Surabaya Gang 1 A No. 71 RT 002 RW 005, Kel. Sumbersari, Kec. Lowokwaru, Kota Malang, Provinsi Jawa Timur.
www.futuresciencepress.com

KATA PENGANTAR

Puji syukur kami panjatkan kepada Tuhan Yang Maha Esa atas rahmat dan karunia-Nya sehingga buku "Algoritma dan Pemrograman dengan C++" ini dapat terselesaikan dengan baik.

Buku Algoritma dan Pemrograman dengan C++ adalah panduan komprehensif yang dirancang untuk membantu pembaca memahami konsep dasar dan implementasi algoritma menggunakan bahasa pemrograman C++. Buku ini mencakup berbagai konsep dasar algoritma dan pemrograman sederhana seperti percabangan, perulangan, array, fungsi, struktur, pointer dan lainnya. Setiap bab memberikan penjelasan mendalam tentang konsep dasar, diikuti dengan contoh kode yang jelas dan praktis dalam C++.

Pembaca akan mempelajari cara menggunakan, mengelola dan memanipulasi data secara efisien menggunakan Bahasa pemrograman C++. Buku ini juga membahas tentang struktur dasar algoritma, Notasi algoritma, dasar dari pemrograman C++. Berdasarkan pendekatan yang terstruktur dan contoh-contoh yang dapat diimplementasikan langsung, buku ini sangat cocok untuk pemula, mahasiswa, dan siapa pun yang ingin memperdalam pengetahuan mereka tentang algoritma dan pemrograman C++. Pada akhirnya, pembaca akan memiliki pemahaman yang kuat tentang bagaimana memecahkan masalah

dengan menggunakan konsep dasar pemrograman berdasar algoritma yang efektif dan efisien dalam C++.

Buku ini diharapkan dapat memberikan kontribusi yang berarti bagi pengembangan ilmu dan praktik dalam bidang ilmu komputer di Indonesia. Kami ingin mengucapkan terima kasih yang sebesar-besarnya kepada Penerbit Future Science yang telah memberikan dukungan dan kesempatan untuk menerbitkan buku ini. Ucapan terima kasih juga kami sampaikan kepada para penulis yang telah berkontribusi dengan pengetahuan, pengalaman, dan dedikasi mereka. Buku ini tidak akan terwujud tanpa kerja keras dan komitmen dari 13 penulis dengan membahas berbagai konsep dasar Algoritma dan Pemrograman.

Akhir kata, kami berharap buku ini dapat memberikan inspirasi dan pengetahuan baru bagi semua yang membacanya. Terima kasih atas perhatian dan dukungannya.

Jakarta, Maret 2025

Editor dan Tim Penulis

DAFTAR ISI

BAB 1
PENGANTAR ALGORITMA DAN PEMROGRAMAN

Norbertus Tri Suswanto Saptadi
Universitas Atma Jaya Makassar, Makassar
E-mail: ntsaptadi@gmail.com

PENDAHULUAN

Algoritma adalah urutan langkah logis yang digunakan untuk memecahkan masalah tertentu. Setiap program komputer pada dasarnya adalah penerapan dari satu atau lebih algoritma (Rangkuti *et al.*, 2023). Oleh karena itu, pemahaman yang kuat tentang algoritma sangat penting untuk menjadi pemrogram yang efektif dan handal. Buku ini akan membahas berbagai jenis algoritma, cara merancang, dan cara mengimplementasikan dalam kode.

Pemrograman merupakan keterampilan yang sangat dibutuhkan era *digital* saat ini. Pengantar ini dirancang untuk memberikan inisiasi yang komprehensif tentang algoritma dan pemrograman yang baru mengenal bidang ini. Pemrograman bukan hanya tentang menulis kode, tetapi juga tentang upaya menyelesaikan persoalan dengan cara yang efisien dan terstruktur (Aisyah and Yahfizham, 2023). Melalui buku ini diharapkan akan belajar cara berpikir seperti seorang pemrogram dan memahami dasar-dasar yang penting dan dibutuhkan untuk meningkatkan kapasitas perangkat lunak agar memiliki nilai dan berkualitas tinggi.

Buku ini akan memperkenalkan konsep dasar pemrograman. Dari variabel dan tipe data hingga struktur kontrol dan fungsi, diharapkan akan mendapatkan suatu pemahaman yang mendalam tentang elemen atau komponen dasar yang membentuk program komputer atau aplikasi (Putro,

Anamisa and Mufarroha, 2019). Dengan menguasai dasar-dasar ini, diharapkan akan *'ready'* untuk mempelajari teknik-teknik pemrograman yang lebih canggih, relevan dengan kebutuhan, dan mulai mengembangkan aplikasi secara mandiri.

SEJARAH PEMROGRAMAN DAN ALGORITMA

Algoritma telah ada jauh sebelum komputer modern ditemukan. Salah satu contoh algoritma tertua adalah Algoritma *Euclid* untuk menemukan faktor persekutuan terbesar, yang ditemukan oleh matematikawan Yunani kuno (Novantara and Apriani, 2021). Dalam konteks pemrograman, algoritma menjadi sangat relevan dengan perkembangan mesin *Turing* pada tahun 1930-an oleh Alan Turing, yang meletakkan dasar bagi teori komputasi dan algoritma modern yang digunakan dan dikembangkan hingga sekarang.

Pemrograman komputer telah mengalami evolusi yang panjang sejak awal mulanya. Pada awalnya, pemrograman dilakukan secara manual dengan menggunakan saklar dan kabel pada komputer generasi pertama. Namun, seiring dengan perkembangan teknologi, maka bahasa pemrograman merupakan ranah tinggi seperti *Fortran*, *COBOL*, dan *Lisp* yang mulai dikembangkan pada tahun 1950-an dan 1960-an dan telah memungkinkan pemrogram untuk menulis kode yang lebih mudah dipahami dan dikelola (Abdi *et al.*, 2024).

Pada tahun 1980-an dan 1990-an, dinamika bahasa pemrograman berorientasi objek seperti C++ dan Java telah membawa paradigma baru dalam pengembangan perangkat lunak. Model data berorientasi objek dinyatakan berkontribusi fleksibilitas yang lebih, kemudahan merevisi program, dan dimanfaatkan luas dalam teknik peranti lunak skala besar (Zain Arif Wildan Sugandi *et al.*, 2022). Pemrograman berorientasi objek memungkinkan peningkatan kapasitas perangkat lunak yang modular dan dapat dipelihara. Dalam dekade terakhir,

memahami cara mendesain solusi yang efisien dan efektif untuk berbagai persoalan komputasi. Penggunaan bahasa pemrograman C++ akan memungkinkan untuk mengimplementasikan algoritma tersebut dalam bentuk kode yang dapat dieksekusi oleh komputer.

Contoh-contoh algoritma yang telah disajikan, mulai dari penjumlahan sederhana hingga penghitungan faktorial dan penggunaan struktur data seperti *linked list*, memberikan gambaran tentang berbagai teknik pemrograman yang esensial. Pemahaman tentang struktur kontrol, seperti *loop* dan *conditional statements*, serta penggunaan fungsi, merupakan aspek penting dalam menulis kode yang modular dan mudah dipahami.

Debugging dan pengujian berupaya menemukan dan memperbaiki kesalahan dalam kode memastikan bahwa program berfungsi sesuai dengan yang diharapkan. Pengujian yang baik melibatkan berbagai teknik, termasuk pengujian unit, integrasi, dan sistem, serta menggunakan alat otomatisasi untuk meningkatkan efisiensi.

Penguasaan algoritma dan pemrograman bukan hanya tentang menulis kode, tetapi juga tentang memahami dan menerapkan konsep dasar yang akan membekali pemrogram dengan kemampuan untuk menangani berbagai tantangan dalam pengembangan perangkat lunak terutama dalam perjalanan menjadi pemrogram yang kompeten dan inovatif.

DAFTAR PUSTAKA

Abdi, F.A. *et al.* (2024) ‘Analisis Perkembangan Dan Klasifikasi Komputer Dari Awal Konsep Hingga Era Modern’, *Nusantara Journal Of Multidisciplinary Science*, 2(1), pp. 14–27. Available at: https://jurnal.intekom.id/index.php/njms/article/view/242/211.

Agustina, S.S. (2022) ‘Analisis Prosedur dan Fungsi dalam Struktur Data pada Sistem Informasi Akademik’, *Jurnal Portal Data*, 2(11), pp. 1–17. Available at: http://portaldata.org/index.php/portaldata/article/view/265.

Aisyah, S. and Yahfizham (2023) ‘Manfaat Pemahaman Algoritma Pemrograman dalam Meningkatkan Kemampuan Pemecahan Masalah’, *Jurnal Arjuna: Publikasi Ilmu Pendidikan, Bahasa dan Matematika*, 1(6).

Alfian, M. (2020) *Buku Ajar Dasar Pemrograman C ++*. Yogyakarta: Lembaga Penerbitan dan Publikasi Ilmiah Universitas Ahmad Dahlan. Available at: http://eprints.uad.ac.id/32726/1/Dasar Pemrograman Bahasa C%2B%2B.pdf.

Anggreani, S. and Yahfizham, Y. (2024) ‘Pengantar dan Pengenalan Konsep Dasar Algoritma Pemrograman’, *Pendekar: Jurnal Pendidikan Berkarakter*, 2(1), pp. 282–294.

Ariya Kusuma, A. and Informatika, T. (2022) ‘Pembahasan Prosedur dan Fungsi pada Pemrograman’, *Portaldata.org*, 2(11), pp. 2022–2023.

Hani Rarti Syahara Harahap; and Yahfizham (2023) ‘Konsep dan Fungsi Algoritma Pemrograman’, *Konstanta : Jurnal Matematika dan Ilmu Pengetahuan Alam*, 1(3), pp. 245–257.

Hasanah, F.N. and Untari, R.S. (2018) ‘Analisis Kemampuan Mendeteksi Eror Kode Program Mata Kuliah Pemrograman Berorientasi Objek pada Program Studi Pendidikan Teknologi Informasi Universitas Muhammadiyah Sidoarjo’, *Teknologi dan Kejuruan: Jurnal Teknologi, Kejuruan, dan Pengajarannya*, 41(2), pp. 139–146.

Indahyanti, U. and Yulianita Rahmawati (2020) *Buku Ajar Algoritma dan Pemrograman dalam Bahasa C++, Buku Ajar Algoritma Dan Pemrograman Dalam Bahasa C++*.

Available at: https://doi.org/10.21070/2020/978-623-6833-67-4.

Khairunnisa, D. (2021) *Buku Ajar Logika dan Algoritma*. Jambi: PT. Sonpedia Publishing Indonesia.

Mohammad Farid Naufal (2018) 'Analisa Teknik Pembelajaran dan Pengajaran pada Universitas dan Industri', *Jurnal Informatika dan Multimedia*, 10(2), pp. 1–8. Available at: https://doi.org/10.33795/jim.v10i2.574.

Mubarak, R. (2020) 'Implementasi Metode White Box Testing pada Proses Quality Assurance Perangkat Lunak Berbasis Web dan Mobile Collection System', *Jurnal Teknologi Informasi ESIT*, 15(1), pp. 57–63.

Mushthofa, D. (2019) *Informatika*. Jakarta: Pusat Kurikulum dan Perbukuan Badan Penelitian dan Pengembangan dan Perbukuan Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi.

Novantara, P. and Apriani, A. (2021) 'Implementasi Algoritma Euclides pada Model Pembelajaran Latihan FPB dan KPK Berbasis Android', *JEJARING : Jurnal Teknologi dan Manajemen Informatika*, 6(2), pp. 43–53.

Praniffa, A.C. *et al.* (2023) 'Pengujian Black Box Dan White Box Sistem Informasi Parkir Berbasis Web Black Box and White Box Testing of Web-Based Parking Information System', *Jurnal Testing dan Implementasi Sistem Informasi*, 1(1), pp. 1–16.

Putro, S.S., Anamisa, D.R. and Mufarroha, F.A. (2019) *Algoritma Pemrograman*. Malang: Media Nusa Creative.

Rangkuti, A. *et al.* (2023) 'Pengenalan Algoritma Pemrograman Dasar dalam Konteks Pembelajaran Pemrograman Awal', *Jurnal Matematika dan Ilmu Pengetahuan Alam*, 1(4), pp. 2987–5315.

Santoso, J.T. (2021) *Struktur Data dan Algoritma, Penerbit Yayasan Prima Agus Teknik*. Available at:

http://penerbit.stekom.ac.id/index.php/yayasanpat/article/view/288.

Saripa, S. (2023) ‘Implementasi Sistem Keamanan File Menggunakan Algoritma AES untuk Mengamankan File Pribadi: Implementasi Sistem Keamanan File Menggunakan Algoritma AES untuk Mengamankan File Pribadi’, *Progressive Information, Security, Computer, and Embedded System*, 1(2), pp. 138–148.

Teknik, J. and Teknik, E.F. (2017) *Operator Modul Praktikum C++ Dasar Pemrograman Komputer*. Malang: Universitas Negeri Malang.

Watrianthos, R. and Purnama, I. (2018) *Buku Ajar Sistem Operasi*. Ponorogo: Uwais Inspirasi Indonesia.

Zain Arif Wildan Sugandi *et al.* (2022) ‘Implementasi Konsep Pemrograman Berorientasi Objek dalam Aplikasi Pembukuan Keuangan Penjual Jus Buah Menggunakan Bahasa Pemrograman Java’, *Jurnal IT CIDA*, 8(1), pp. 1–8.

Zein, Afrizal, & E.S.E. (2022) *Algoritma dan Struktur Data, Fakultas Komputer*. Banten: Unpam Press.

PROFIL PENULIS

Norbertus Tri Suswanto Saptadi

Lahir di Cirebon Jawa Barat, tanggal 7 Juni 1975. Memiliki Jabatan Fungsional Lektor Kepala, Pembina Tingkat I (IV/b). Berpendidikan Sarjana Komputer (S.Kom.) di Universitas Teknologi Digital Indonesia (UTDI) tahun 1998, Magister Manajemen (M.M.) di Universitas Hasanuddin (UNHAS) tahun 2004, Magister Teknologi Informasi (M.T.) di Universitas Gadjah Mada (UGM) tahun 2007, Insinyur (Ir.) di Pendidikan Profesi Insinyur UNHAS tahun 2020, Insinyur Profesional Madya (IPM.) di Persatuan Insinyur Indonesia (PII) tahun 2021, Doktor (Dr.) di Fakultas Teknik UNHAS tahun 2023, dan Program

Pendidikan Reguler Angkatan (PPRA) LX Lemhannas RI tahun 2020. Menjadi tenaga pengajar (Dosen) pada Program Studi Teknik Informatika Fakultas Teknologi Informasi Universitas Atma Jaya Makassar (UAJM). Peraih Poster terbaik DPRM Dikti tahun 2016. Dosen berprestasi IKDKI tahun 2020 dan 2021. Pernah menjabat Kepala UPT Komputer, Kepala BAPSI, Wakil Dekan FT, Dekan FT dan FTI, Wakil Rektor III, Ketua Penjaminan Mutu. Tim PAK Dosen dan Asesor BKD UAJM. Reviewer International Conference dan Jurnal SINTA. Pemenang Hibah Kemdikbud Penelitian Dosen Pemula, Bersaing, Fundamental, dan Strategi Nasional. Penulis artikel media massa Tribun Timur, Koinonia, Bisnis Sulawesi, Sesawi.net, Mirifica.net, HidupKatolikCom, OMKNet, KatolikanaTV, Jalan Hidup Katolik, dll. Penulis Buku di Penerbit Kanisius, Sada Kurnia Pustaka, Aksara Sastra Media, Future Science, HEI Publishing, Mifandi Mandiri Digital, Rey Media Grafika, Widina Salemba, dan Cendikia Mulia Mandiri. Aktifis organisasi IKA Lemhannas RI LX, IARMI, DPP ISKA, BAPOMI Sulsel, LP3KD Sulsel, IKDKI SulSelTraBar, Komkep KAMS, Komsos KAMS, PUKAT KAMS, TPP KAMS, FMKI KAMS, UPS KAMS, Pengurus Kebun Sawit Laimbo, FDI, PII Makassar, INAPR, Dewan Keuangan Paroki dan Program Ayo Sekolah Mariso, Animator Laudato Si', dan lain-lain.

BAB 2
STRUKTUR DASAR ALGORITMA

Ma'shum Abdul Jabbar
Universitas Djuanda, Bogor
E-mail: mashum.jabbar@unida.ac.id

PENDAHULUAN

Algoritma merupakan suatu langkah-langkah atau prosedur yang sistematis dalam memecahkan suatu masalah, dan seringkali digunakan dalam berbagai konteks, seperti salah satu contohnya adalah dalam pengelolaan energi listrik ataupun penjadwalan praktikum, yang memiliki kompleksitas dan kebutuhan yang berbeda-beda.

Siallagan dan Susantok (2021) mengembangkan sistem kontrol dan monitoring energi pada rumah pintar yang menggunakan algoritma percabangan bersyarat untuk memungkinkan perangkat IoT mengambil keputusan berdasarkan data yang dihasilkan. Sistem ini menggunakan algoritma percabangan bersyarat memungkinkan sistem untuk menilai kondisi tertentu dan mengambil keputusan berdasarkan parameter yang ada, sehingga menjadikan sistem ini lebih pintar dan efisien.

Roihan et al. (2022) merancang sebuah aplikasi penjadwalan praktikum laboratorium dengan menerapkan algoritma greedy yang digabungkan dengan perulangan untuk mengoptimalkan pembuatan jadwal. Dengan menggunakan algoritma ini, pembuatan jadwal praktikum menjadi lebih terstruktur dan tidak memerlukan waktu yang lama.

Dari kedua contoh tersebut, dapat dilihat bahwa algoritma digunakan dalam berbagai domain untuk meningkatkan efisiensi dan pengambilan keputusan yang lebih baik.

DEFINISI DAN PENTINGNYA ALGORITMA

Algoritma merupakan suatu prosedur komputasi yang didefinisikan dengan mengambil suatu nilai sebagai input dan menghasilkan suatu nilai sebagai output. Algoritma merupakan metode atau suatu rangkaian proses yang bisa dipahami oleh manusia dan digunakan untuk pemecahan masalah. Selanjutnya, jika proses tersebut dikerjakan oleh komputer maka instruksi tersebut perlu disusun ke dalam bahasa yang dipahami oleh komputer atau dikenal sebagai bahasa pemrograman.

Kemajuan ilmu pengetahuan dan teknologi memungkinkan manusia menciptakan karya-karya yang semakin canggih dan kompleks. Walaupun komputer mampu melakukan perhitungan lebih cepat dibandingkan manusia, komputer tidak dapat menyelesaikan masalah secara otomatis tanpa terlebih dahulu diberikan instruksi berupa langkah-langkah penyelesaian (algoritma) yang dirancang oleh manusia. Algoritma tidak hanya berguna untuk menyelesaikan masalah dengan komputer, tetapi juga dapat diterapkan dalam kehidupan sehari-hari untuk memecahkan masalah yang memerlukan proses atau langkah-langkah prosedural. Untuk memahami algoritma lebih mendalam, mari kita pelajari definisinya dari berbagai sumber.

Pada beberapa sumber diperoleh pengertian dari algoritma antara lain:

1. Kata "algoritma" berasal dari istilah *algoris* dan *ritmis*, yang pertama kali diperkenalkan oleh Abu Ja’far Muhammad Ibn Musa Al Khwarizmi pada tahun 825 M melalui karyanya *Al-Jabr Wa-al Muqabla*. Dalam bidang pemrograman, algoritma didefinisikan sebagai metode yang terdiri dari serangkaian langkah yang terstruktur dan sistematis untuk menyelesaikan masalah dengan bantuan komputer (Jando & Nani, 2018, hlm. 5).
2. Menurut Kani (2020, hlm. 1.19), algoritma adalah serangkaian operasi yang dirancang secara logis dan

loncatan. Struktur pemilihan memungkinkan algoritma untuk mengambil keputusan berdasarkan kondisi tertentu, menjadikannya fleksibel dalam menangani berbagai situasi. Sementara itu, struktur perulangan memungkinkan eksekusi berulang terhadap serangkaian langkah sampai kondisi tertentu terpenuhi, sehingga sangat berguna dalam mengolah data yang besar atau tugas yang membutuhkan iterasi. Ketiga struktur ini saling melengkapi, membentuk fondasi logika pemrograman yang mampu menyelesaikan masalah dengan solusi yang optimal.

DAFTAR PUSTAKA

GeeksforGeeks. (2021, June 30). *Decision Making in C / C++ (if , if..else, Nested if, if-else-if)*. GeeksforGeeks. Retrieved December 10, 2021, from https://www.geeksforgeeks.org/decision-making-c-c-else-nested-else/

Hasan, M., & Yahfizham, Y. (2024). Pengenalan Algoritma pada Pembelajaran Pemrograman Komputer. Comit: Communication, Information and Technology Journal, 2(2), 355-368.

Jando, E., & Nani, P. A. (2018). *Algoritma dan Pemrograman dengan Bahasa Java*. Penerbit ANDI.

Kani. (2020). *Algoritma dan Pemrograman*. Universitas Terbuka.

Muhardian, A. (2021, December 2). *Belajar Java: Memahami 3 Bentuk Percabangan dalam Java*. Petani Kode. Retrieved December 10, 2021, from https://www.petanikode.com/java-percabangan/

Purnamasari, P. (2021). Teori Atau Konsep Algoritma Pemrograman.

Roihan, A., Nasution, K., & Siambaton, M. Z. (2022). Implementasi Algoritma Greedy Kombinasi dengan

Perulangan pada Aplikasi Penjadwalan Praktikum. *Sudo Jurnal Teknik Informatika*, *1*(2), 42–50. https://doi.org/10.56211/sudo.v1i2.8

Siallagan, S., & Susantok, M. (2021). Implementasi Wise Smart Home Untuk Penggunaan Energi Menggunakan Algoritma Percabangan Bersyarat Pada Server Blynk. *ABEC Indonesia*, *9*, 983–998.

Virk, M. (2024, May 17). *Programming Essentials Unveiled: The Art of Sequences, Selections, and Loops*. Skool of Code. Available at: https://skoolofcode.us/blog/programming-essentials-unveiled-the-art-of-sequences-selections-and-loops/. Accessed December 2, 2024.

PROFIL PENULIS

Ma'shum Abdul Jabbar, S.Kom., M.T.I.

Penulis memiliki latar belakang pendidikan Magister Teknik Informatika dari Universitas Bina Nusantara pada Tahun 2020, Sarjana Teknik Informatika Universitas MH Thamrin pada Tahun 2018 dan Diploma III Teknik Informatika Politeknik Kampar pada Tahun 2015. Penulis cukup sering mengikuti berbagai pelatihan di rumpun ilmu IT dan telah memiliki beberapa sertifikasi global termasuk sertifikasi PCAP – Certified Associate in Python Programming pada tahun 2020. Penulis juga memiliki pengalaman bekerja sebagai praktisi IT sejak tahun 2015 dengan berbagai posisi di beberapa perusahaan yang bergerak di bidang seperti Perusahaan telekomunikasi, pemerintah daerah, start up education technology dan perusahaan konsultan IT. Saat ini penulis juga menjadi pengajar pada program studi ilmu komputer di salah satu perguruan tinggi swasta di daerah Bogor serta beberapa kali menjadi narasumber pada acara seminar. Sebagai seorang penulis dan praktisi IT, penulis ingin berbagi wawasan terhadap keilmuan di bidang Programming yang dapat membantu pembaca memahami materi seputar Algoritma dan Pemrograman.

BAB 3
NOTASI ALGORITMA BERDASARKAN KALIMAT DESKRIPTIF, FLOWCHART, DAN PSEUDOCODE

Muhammad Dedi Irawan
Universitas Islam Negeri Sumatera Utara, Medan
E-mail: md.irawan@uinsu.ac.id

PENDAHULUAN

Algoritma adalah landasan dari pengembangan perangkat lunak, karena menentukan urutan langkah-langkah yang diperlukan untuk menyelesaikan suatu masalah atau tugas secara efisien dan efektif. Penggunaan notasi yang tepat dalam penyusunan algoritma sangat penting karena membantu pengembang perangkat lunak dan pemangku kepentingan lainnya untuk memahami, memvalidasi, dan mengimplementasikan algoritma tersebut dengan benar. Di dalam dunia pemrograman, terdapat tiga notasi utama yang sering digunakan untuk merepresentasikan algoritma, yaitu notasi deskriptif, flowchart, dan pseudocode.

Notasi deskriptif adalah cara yang paling sederhana dan intuitif untuk mendokumentasikan algoritma, karena menggunakan bahasa sehari-hari untuk menjelaskan langkah-langkah yang harus diambil. Ini membuatnya sangat berguna, terutama dalam tahap awal pembelajaran pemrograman, di mana fokus utamanya adalah pada pemahaman logika dasar tanpa perlu khawatir tentang sintaksis yang rumit. Meskipun mudah dipahami, notasi deskriptif memiliki keterbatasan dalam hal struktur dan presisi, yang dapat menjadi masalah ketika berurusan dengan algoritma yang lebih kompleks.

Flowchart, di sisi lain, menawarkan cara visual untuk merepresentasikan algoritma. Dengan menggunakan simbol-

simbol standar seperti oval, kotak, dan panah, flowchart memungkinkan pengembang untuk melihat secara visual bagaimana langkah-langkah dalam algoritma saling berhubungan. Ini sangat berguna dalam tahap perencanaan dan analisis sistem, karena memberikan gambaran yang jelas tentang alur proses yang akan diikuti oleh program. Namun, flowchart sering kali memerlukan ruang yang lebih besar dan bisa menjadi rumit untuk algoritma yang kompleks.

Pseudocode adalah bentuk perantara antara bahasa manusia dan bahasa pemrograman. Ini tidak terikat oleh aturan sintaks yang ketat seperti bahasa pemrograman, tetapi cukup terstruktur untuk diterjemahkan langsung menjadi kode program. Pseudocode memungkinkan pengembang untuk fokus pada logika algoritma tanpa harus memikirkan detail-detail teknis dari bahasa pemrograman tertentu. Ini membuatnya sangat berguna dalam dokumentasi dan komunikasi antar programmer yang menggunakan berbagai bahasa pemrograman.

Pemahaman tentang ketiga notasi ini, beserta kelebihan dan kekurangannya, sangat penting bagi siapa pun yang terlibat dalam pengembangan perangkat lunak. Dalam praktiknya, seringkali diperlukan kombinasi dari ketiga notasi tersebut untuk memastikan bahwa algoritma dapat diimplementasikan dengan efektif dan efisien.

NOTASI ALGORITMA BERDASARKAN KALIMAT DESKRIPTIF

Notasi algoritma berbasis kalimat deskriptif adalah metode sederhana yang menjelaskan langkah-langkah algoritma menggunakan bahasa sehari-hari, membuatnya mudah dipahami bahkan oleh orang tanpa latar belakang teknis. Metode ini sangat bermanfaat dalam tahap awal pengajaran pemrograman, di mana fokusnya adalah memahami logika dasar tanpa kompleksitas sintaksis pemrograman. Konsep ini sering dibahas

Dalam pengembangan perangkat lunak yang kompleks, sering kali dibutuhkan kombinasi dari ketiga notasi tersebut. Flowchart dapat digunakan pada tahap perencanaan untuk memberikan gambaran umum tentang alur proses, sementara pseudocode digunakan pada tahap desain dan implementasi untuk memastikan bahwa logika algoritma dapat diterapkan dalam kode program. Dengan pemahaman yang baik tentang kapan dan bagaimana menggunakan setiap notasi, pengembang perangkat lunak dapat memastikan bahwa algoritma yang mereka buat tidak hanya efektif dalam menyelesaikan masalah, tetapi juga efisien dan mudah dipahami oleh orang lain.

DAFTAR PUSTAKA

Chaudhuri, A. B. (2020). *Flowchart and algorithm basics: The art of programming*. Mercury Learning and Information.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to algorithms* (Fourth edition). The MIT Press.

Greenaway, J. (2023). *FUNDAMENTALS FOR SELF-TAUGHT PROGRAMMERS embark on your software engineering journey without exhaustive courses and bulky tutorials* (1st edition). PACKT PUBLISHING LIMITED.

Irawan, M. D. (2022). *Flowchart dan Pseudo-Code: Implementasi Notasi Algoritma dan Pemrograman*. Media Sains Indonesia.

Janfada, A. S. (2019). *Elementary Synchronous Programming: In C++ and Java via algorithms*. De Gruyter. https://doi.org/10.1515/9783110616484

Jaton, F. (2021). *The constitution of algorithms: Ground-truthing, programming, formulating*. The MIT Press.

Knuth, D. E. (2021). *The art of computer programming, Volume 4A: Combinatorial algorithms, Part 1*. Addison-Wesley.

Rinaldi Munir, & Lidya, L. (2016). *Algoritma dan Pemrograman Dalam Bahasa Pascal, C, dan C++* (Edisi Keenam). Informatika.

PROFIL PENULIS

Muhammad Dedi Irawan, S.T., M.Kom.
Penulis adalah seorang akademisi dan peneliti di bidang Artificial Intelligence, Decision Support System, Information Technology, dan Computer Science. Saat ini, beliau mengajar di Universitas Islam Negeri Sumatera Utara. Dalam peran ini, beliau berkontribusi dalam penelitian dan pengembangan teknologi, terutama yang terkait dengan sistem pendukung keputusan dan kecerdasan buatan. Beliau juga aktif mempublikasikan karya ilmiah di berbagai, buku, artikel serta jurnal internasional dan konferensi akademis.

BAB 4
PENGANTAR BAHASA PEMROGRAMAN C++

Erna Hudianti Pujiarini
Universitas Teknologi Digital Indonesia, Yogyakarta
E-mail: ernahudi@utdi.ac.id

SEJARAH DAN PERKEMBANGAN BAHASA C++

Bahasa pemrograman C++ dikembangkan oleh Bjarne Stroustrup di Bell Laboratories, Amerika Serikat, pada awal 1980-an (Grimm, 2020). Stroustrup, seorang ilmuwan komputer kelahiran Denmark, menciptakan C++ sebagai solusi untuk mengatasi kekurangan bahasa pemrograman lain yang ada pada saat itu, terutama dalam konteks efisiensi dan fleksibilitas. C++ pada dasarnya adalah perluasan dari bahasa pemrograman C, yang sangat populer karena kemampuannya mengelola perangkat keras secara langsung dan performanya yang cepat. Namun, Stroustrup menginginkan bahasa yang tidak hanya seefisien C, tetapi juga mendukung konsep pemrograman berorientasi objek (Object-Oriented Programming, atau OOP).

Stroustrup terinspirasi oleh bahasa Simula, yang dikembangkan pada 1960-an di Norwegia. Simula merupakan bahasa pemrograman pertama yang memperkenalkan konsep kelas dan objek, yang memungkinkan pemrogram untuk mengorganisasikan kode berdasarkan entitas yang mewakili objek dalam dunia nyata. Pemrograman berorientasi objek sangat membantu dalam mengelola program besar karena memungkinkan pengembangan kode yang lebih modular dan dapat digunakan kembali. Meski Simula memperkenalkan pendekatan yang revolusioner, bahasa ini dianggap lambat dalam eksekusi, yang membuatnya tidak ideal untuk aplikasi yang membutuhkan performa tinggi.

Stroustrup mulai bekerja pada bahasa yang diberi nama “C with Classes” pada tahun 1979. Ide utama di balik "C with Classes" adalah menggabungkan efisiensi dan fleksibilitas C dengan konsep-konsep OOP dari Simula. Ia menambahkan fitur-fitur seperti kelas, enkapsulasi, pewarisan, dan polimorfisme ke dalam bahasa C, sehingga membuatnya mampu menangani masalah-masalah yang lebih kompleks dengan cara yang lebih terstruktur. Stroustrup juga memastikan bahwa bahasa ini tetap kompatibel dengan C, memungkinkan program-program yang ditulis dalam C dapat berjalan dalam "C with Classes" tanpa perubahan besar.

Pada tahun 1983, Stroustrup memutuskan untuk mengubah nama "C with Classes" menjadi C++. Nama ini diambil dari operator ++ dalam bahasa C, yang berarti "increment" atau "penambahan satu". Filosofinya, C++ adalah bahasa C yang ditingkatkan, menambahkan kemampuan-kemampuan baru tanpa kehilangan karakteristik yang telah membuat C populer. Nama baru ini juga menandai dimulainya era baru dalam pengembangan perangkat lunak.

Salah satu fitur pertama yang Stroustrup tambahkan adalah overloading operator. Dalam C++, operator seperti +, -, dan * dapat didefinisikan ulang untuk bekerja dengan tipe data yang dibuat oleh pengguna (seperti kelas). Ini membuat C++ semakin fleksibel dan sangat berguna untuk aplikasi yang memerlukan representasi matematis kompleks, seperti perangkat lunak teknik dan ilmiah. Selain itu, Stroustrup juga memperkenalkan multiple inheritance ke dalam bahasa C++, yang memungkinkan sebuah kelas mewarisi atribut dan perilaku dari lebih dari satu kelas induk. Ini menjadi keunggulan C++ yang tidak tersedia di banyak bahasa lain pada waktu itu.

Selama tahun 1980-an, bahasa C++ mulai diadopsi oleh industri dan komunitas akademik. Banyak pengembang yang tertarik dengan kemampuan OOP yang ditawarkan C++, serta

b. Komentar banyak baris

Cara menyatakan komentar banyak baris dengan menggunakan /* untuk membuka dan */ untuk menutup komentar. Cocok untuk penjelasan yang panjang.

Contoh :

```
/* Program ini menghitung luas lingkaran
Menggunakan rumus: luas = π * r * r   */
float luas = 3.14 * radius * radius;
```

Berikut ini adalah contoh lengkap program C++ sederhana :

```
#include <iostream>      // Menyertakan pustaka iostream untuk input dan output

using namespace std;      // Menggunakan namespace standar untuk menghindari penulisan std::

int main() {              // Fungsi utama yang menjadi titik awal program
   cout << "Selamat siang" << endl; // Menampilkan " Selamat siang " ke layar
   return 0;              // Mengakhiri program dan mengembalikan nilai 0
}
```

Penjelasan contoh program diatas:

- Perintah #include <iostream> digunakan untuk menyertakan pustaka untuk melakukan input-output.
- Perintah using namespace std; digunakan untuk mempermudah penggunaan elemen standar.
- Perintah int main() { ... } merupakan fungsi utama (main) di mana eksekusi program dimulai.
- Perintah cout << " Selamat siang " << endl; digunakan untuk menampilkan teks " Selamat siang " di layar.

- Perintah return 0; digunakan untuk mengakhiri eksekusi program dan mengembalikan nilai 0 ke sistem operasi, yang biasanya menandakan bahwa program selesai tanpa kesalahan.

DAFTAR PUSTAKA

Grimm, R. (2020). *C++20: Get the Details.* -: Leanpub.

Malik, D. (2018). *C++ Programming: From Problem Analysis to Program Design.* Boston: Cengage Learning.

Stroustrup, B. (2014). *Programming: Principles and Practice Using C++ (2nd Edition).* Indiana: Addison-Wesley.

PROFIL PENULIS

Erna Hudianti Pujiarini, S.Si, M.Si.

Lahir di Magetan, 28 September 1971. Lulus Sarjana di Program Studi Statistik Universitas Gadjah Mada tahun 1990, lulus program studi Matematika Program Pasca Sarjana Universitas Gadjah Mada tahun 2000. Mulai mengajar tahun 1996 di STMIK AKAKOM yang sekarang berubah menjadi Universitas Teknologi Digital Indonesia. Saat ini menjadi dosen pada program studi Informatika, fakultas Teknologi Informasi, Universitas Teknologi Digital Indonesia. Matakuliah yang diampu Statistika, Statistika Terapan, Matematika, Data Mining, Jaringan Syarat Tiruan. Bidang konsentrasi penelitian Statistika Terapan dan Data Science. Buku yang ditulis “SPSS: Analisis Data Statistik”, “Data Mining : Pola dibalik Angka”, “Statistika Dasar”, “Dasar Pemrograman dengan Bahasa Pemrograman Python”.

BAB 5
TIPE DATA DAN OPERATOR

Mohammad Badrul
Universitas BSI, Jakarta
E-mail: mohammad.mbl@bsi.ac.id

PENDAHULUAN

Bahasa Pemrograman C++ merupakan salah satu bahasa pemrograman dasar yang banyak digunakan oleh kalangan siswa atau mahasiswa untuk memulai belajar bahasa Pemrograman. Salah satu materi yang perlu dipahami adalah tipe data, deklarasi variabel dan operator. Tipe data menentukan bagaimana data disimpan di memori komputer dan bagaimana data tersebut dapat diakses atau dimanipulasi. Tipe data tidak hanya menggambarkan jenis informasi yang disimpan, tetapi juga mempengaruhi lokasi penyimpanan dan penggunaan memori. Setiap tipe data memiliki rentang nilai tertentu, menentukan jenis operasi yang dapat dilakukan pada suatu variabel, memilih tipe data yang tepat membantu mencegah bug dan kesalahan. Identifier di bahasa Pemrograman C++ terdiri dari nama fungsi, variabel, array, struktur, konstanta, procedure, queue atau nama pointer. Selain pemahaman identifier dan tipe data, pemahaman operator di Pemrograman C++ perlu dipahami seperti operator aritmatika, operator penugasan, operator logika dan operator perbandingan.

TIPE DATA

Merupakan kategori atau jenis data untuk menyimpan dan memproses informasi dalam suatu program atau sistem. Setiap tipe data memiliki karakteristik dan fungsi tertentu (Weiss, 2013). Dalam pemrograman, tipe data menentukan bagaimana

data disimpan di memori komputer dan bagaimana data tersebut dapat diakses atau dimanipulasi. Tipe data tidak hanya menggambarkan jenis informasi yang disimpan, tetapi juga mempengaruhi lokasi penyimpanan dan penggunaan memori(Jain, 2018). Penentuan tipe data di C++ penting karena beberapa alasan mendasar seperti mempengaruhi bagaimana program bekerja secara efisien dan akurat, menentukan jumlah memori yang akan dialokasikan untuk suatu variabel, Setiap tipe data memiliki rentang nilai tertentu, menentukan jenis operasi yang dapat dilakukan pada suatu variabel, memilih tipe data yang tepat membantu mencegah bug dan kesalahan, Kompatibilitas dengan Fungsi atau Struktur Data dan Kompatibilitas Antar Sistem.

Dalam bahasa pemrogaman C++ terdapat beberapa tipe data dasar yang dapat digunakan antara lain(Sianipar, 2017):

1. Tipe data integer

   Merupakan tipe data yang digunakan untuk menyimpan nilai numerik berupa bilangan bulat (tanpa desimal). Tipe data ini umum digunakan di berbagai bahasa pemrograman untuk merepresentasikan nilai seperti hitungan, indeks, atau angka yang tidak memerlukan presisi desimal.

   Misalnya 4, 12, 16, 26.

   Bentuk umum penulisannya adalah

   ```
   int namaVariabel;
   ```

   Contoh Deklarasi variabel

   ```
   int nilai = 90;
   ```

   Contoh aplikasi sederhana menggunakan aplikasi Dev C++

Pemrograman. Salah satu materi pembahasan yang harus dipahami adalah memahami tipe data dan variabel karena jika salah melakukan deklarasi variabel atau tipe data bukan tidak mungkin aplikasi yang dibuat akan error. Kalaupun bisa running, hasil yang diperoleh tidak akan sesuai dengan yang diharapakan. Selain memahami variabel dan tipe data, hal yang perlu dipahami adalah operator. Ada 4 jenis operator yang perlu dipahami adalah operator penugasan, operator aritmatika, operator logika dan operator perbandingan sesuai dengan kebutuhan atau kasus yang akan dikerjakan.

DAFTAR PUSTAKA

Chairunnisa, N. (2024). *Langkah Mudah Belajar Pemrograman C++ untuk Pemula*. Anak Hebat Indonesia.

Guntara, R. G. (2023). *Algoritma Pemrograman Dasar : Menggunakan Bahasa Pemrograman C++ dengan Contoh Kasus Aplikasi untuk Bisnis dan Manajemen*. CV. Ruang Tentor.

Jain, H. (2018). *Problem Solving in Data Structures & Algorithms Using C++: Programming Interview Guide First Edition*. CreateSpace Independent.

Kadir, A. (2012). *Buku pintar c++ untuk pemula : panduan praktis mempelajari pemrograman dengan cepat, mudah dan menyenangkan*. MediaKom.

Sianipar, R. H. (2017). *Teori dan Aplikasi C++ dengan Contoh Lebih dari 280 Source Code*. Andi Publisher.

Sindar, A. (2019). *Struktur Data dan Algoritma dengan C++*. CV. AA Rizky.

Weiss, M. (2013). *Data Structures & Algorithm Analysis in C++* (4th ed.). Pearson.

PROFIL PENULIS

Mohammad Badrul

Menyelesaikan Study Strata 1 (S1) di Kampus STMIK Nusa Mandiri dengan Jurusan Sistem Informasi dan menyelesaikan program Srata 2 (S2) di Kampus yang sama dengan jurusan ilmu Komputer. Saat ini penulis aktif mengajar bidang komputer di beberapa kampus di Jakarta seperti di Universitas Bina Sarana Informatika, Universitas Darma Persada, Institute Bisnis dan Informatika Kwik Kian Gie, Universitas Dian Nusantara dan Universita Terbuka. Selain mengajar, Penulis aktif juga melakukan penelitian khususnya di bidang data mining, machine learning, Jaringan komputer, Information System dan Decision Support System. Penulis juga Aktif dalam membimbing mahasiswa yang sedang melakukan penelitian khususnya di tingkat strata 1.

BAB 6
FUNGSI INPUT DAN OUTPUT PADA C++

Susi Erlinda
Universitas Sains dan Teknologi Indonesia, Pekanbaru Riau
E-mail: susierlinda@usti.ac.id

PENDAHULUAN

Dalam pemrograman C++, fungsi input dan output (I/O) adalah salah satu komponen kunci yang memungkinkan komunikasi antara aplikasi dan pengguna serta sistem. Fungsi I/O mendukung berbagai jenis data dan format, serta menyediakan fleksibilitas dalam penanganan data. Bab ini membahas penerapan fungsi I/O dalam C++, mencakup penggunaan dasar, manipulasi format, penanganan kesalahan, dan teknik lanjutan seperti file I/O dan streaming.

DASAR-DASAR FUNGSI I/O DI C++

STREAM DALAM C++

Di C++, *stream* adalah objek yang digunakan untuk komunikasi data antara aplikasi dan perangkat I/O. Stream mempermudah operasi input dan output dengan menyediakan metode yang konsisten untuk membaca dan menulis data. Ada dua jenis stream utama:

a. **Input Stream (std::cin)**: Digunakan untuk membaca data dari perangkat input, seperti keyboard.
b. **Output Stream (std::cout)**: Digunakan untuk menampilkan data ke perangkat output, seperti layar monitor.

cpp
Salin kode
#include <iostream>

```
using namespace std;
int main() {
cout << "Hello, World!" << endl;
return 0;
}
```

I/O berbasis stream di C++ menawarkan pendekatan yang terpadu untuk mengelola data, memberikan kemudahan penggunaan serta fleksibilitas (Gabbard & Roberts, 2023).

HEADER DAN NAMESPACE

Untuk menggunakan fungsi I/O, Anda harus menyertakan header <iostream> dan menggunakan namespace std:

cpp
Salin kode

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

Header ini menyediakan objek stream standar seperti std::cin, std::cout, dan std::cerr.

FUNGSI INPUT DAN OUTPUT DASAR
MENAMPILKAN OUTPUT DENGAN STD::COUT

Fungsi std::cout digunakan untuk menampilkan output ke layar. Output bisa berupa teks, variabel, atau hasil perhitungan:

cpp
Salin kode

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

int main() {
int age = 25;
cout << "I am " << age << " years old." << endl;
return 0; }
```

KESIMPULAN

Pemahaman mendalam tentang fungsi input dan output di C++ sangat penting untuk pengembangan aplikasi yang efektif dan interaktif. Dari penggunaan dasar stream hingga teknik lanjutan seperti file I/O dan manipulasi format, C++ menyediakan berbagai alat untuk mengelola data dengan efisien. Praktik dan eksperimen lebih lanjut akan meningkatkan keterampilan dalam menggunakan fungsi I/O ini.

DAFTAR PUSTAKA

Lippman, S. B., Lajoie, J., & Moo, B. E. (2023). *C++ Primer* (6th ed.). Addison-Wesley.

Gabbard, J., & Roberts, S. (2023). *Advanced C++ Programming by Example*. Apress.

Joshi, A. (2022). *Mastering C++ Programming*. Packt Publishing.

Eubanks, M. (2023). *Practical C++ Programming: From Beginner to Expert*. O'Reilly Media.

Griessenberger, K. (2023). *C++ in Action: Real-world Projects and Techniques*. Manning Publications.

Moo, B. E. (2022). *Effective Modern C++: 42 Specific Ways to Improve Your Use of C++11 and C++14*. Addison-Wesley.

Vandevoorde, D., & Josuttis, N. M. (2022). *C++ Templates: The Complete Guide* (2nd ed.). Addison-Wesley.

Plum, T. (2022). *The C++ Programming Language* (4th ed.). Addison-Wesley.

BAB 7
PERCABANGAN PADA C++

Yusuf Ramadhan Nasution
Universitas Islam Negeri Sumatera Utara, Medan, Indonesia
E-mail: ramadhannst@uinsu.ac.id

PENDAHULUAN

Percabangan merupakan salah satu elemen penting dalam pemrograman, termasuk dalam bahasa C++. Dengan percabangan, program dapat membuat keputusan berdasarkan kondisi tertentu. Ini memungkinkan program berjalan lebih fleksibel dan responsif terhadap input atau data tertentu. Dalam dunia programming, algoritma percabangan dipakai dalam berbagai macam skenario. Misalnya, pemilihan opsi pembayaran pada suatu aplikasi atau scoring test filtering otomatis (Rus'an, 2022).

Percabangan adalah blok program yang digunakan untuk menentukan aksi mana yang akan dieksekusi tergantung benar atau tidaknya kondisi yang didefinisikan. Suatu percabangan selalu melibatkan kondisi yaitu ekspresi logika yang bisa bernilai benar atau salah. Suatu aksi yang ditempatkan dalam blok pemilihan hanya akan dieksekusi jika kondisi yang didefinisikan bernilai benar atau syarat yang diminta terpenuhi. Kondisi harus selalu diapit dengan tanda kurung. Didalam C++ blok percabangan bisa dibuat dengan menggunakan perintah if. Percabangan akan mampu membuat program berpikir dan menentukan tindakan sesuai dengan logika/kondisi yang kita berikan (Muhardian, 2018).

Percabangan dalam C++ adalah struktur kontrol program yang memungkinkan pengambilan keputusan berdasarkan kondisi tertentu. Dengan percabangan, program dapat

menjalankan bagian kode yang berbeda sesuai dengan hasil evaluasi suatu kondisi logika. Materi ini membahas berbagai bentuk percabangan, seperti if, if-else, if-else if, dan switch, yang digunakan untuk menangani berbagai situasi. Percabangan akan mampu membuat program berpikir dan menentukan tindakan sesuai dengan logika/kondisi yang kita berikan. Percabangan If digunakan saat terdapat satu pilihan Keputusan (Taufik Hidayat, ST., 2023).

JENIS-JENIS PERCABANGAN

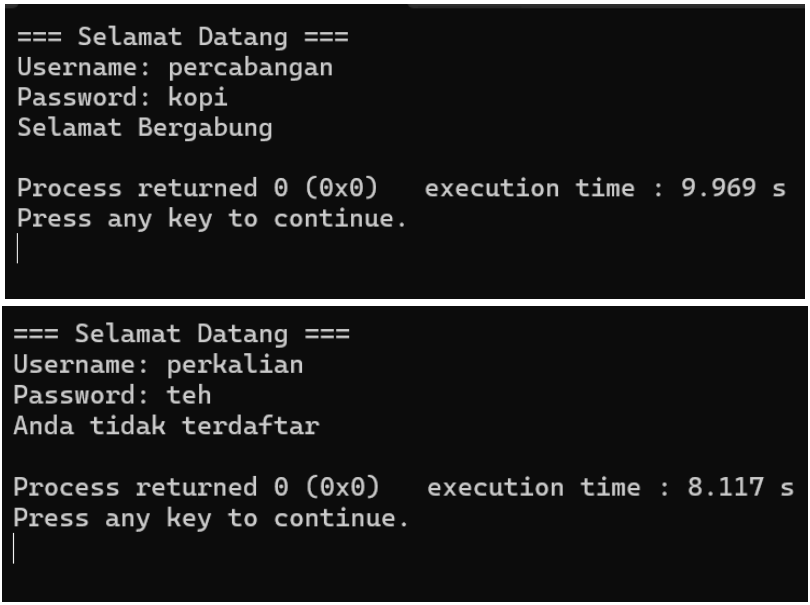
1. Percabangan *IF*

Percabangan IF adalah salah satu struktur kontrol dalam C++ yang digunakan untuk menjalankan blok kode tertentu hanya jika suatu kondisi terpenuhi (bernilai benar atau true). Jika kondisi tersebut salah atau false, maka blok kode tersebut akan dilewati, dan program melanjutkan ke pernyataan berikutnya.

Percabangan if digunakan untuk membuat program yang dapat mengambil keputusan berdasarkan kondisi tertentu, sehingga alur eksekusinya menjadi dinamis.

Berikut contoh flowchart percabangan *IF*:

Contoh output program saat dijalankan:



Gambar 7.16. Output Program *Nested*

KESIMPULAN

Percabangan pada C++ adalah dasar untuk logika pengambilan keputusan, yang berperan penting dalam membuat program lebih interaktif dan adaptif. Percabangan mencakup struktur logika dalam C++ yang memungkinkan program mengambil keputusan berdasarkan kondisi tertentu. Dengan fitur seperti if, if-else, switch-case, operator ternary (? :), dan nested conditional, programmer dapat menangani berbagai skenario logis dengan fleksibilitas dan efisiensi.

Dengan memahami jenis-jenis percabangan dan cara penggunaannya secara efektif, programmer dapat menciptakan program yang efisien, mudah dikelola, dan mampu menangani berbagai situasi dengan baik.

DAFTAR PUSTAKA

Muhardian, A. (2018). *Belajar C++ #07: Memahami 6 Macam Bentuk Blok Percabangan pada C++*. Petanikode. https://www.petanikode.com/cpp-percabangan/

Rus'an, Z. E. (2022). *Tiga Jenis Algoritma yang Perlu Kamu Ketahui*. Dicoding. https://www.dicoding.com/blog/tiga-jenis-algoritma-yang-perlu-kamu-ketahui/

Taufik Hidayat, ST., M. (2023). *Matakuliah Algoritma dan Pemrograman Semester Ganjil 2022 / 2023*.

PROFIL PENULIS

Yusuf Ramadhan Nasution, M.Kom

Penulis adalah seorang dosen dan praktisi di bidang Teknologi Informasi dengan pengalaman lebih dari 10 tahun. Lahir di Medan, 25 Mei 1985. Menyelesaikan pendidikan S1 Teknik Informatika di STMIK Budidarma Medan dan meraih gelar Magister Teknologi Informasi dari Universitas Putra Indonesia "YPTK" Padang. Kepakaran penulis meliputi pengembangan perangkat lunak, sistem basis data, dan kecerdasan buatan. Selain mengajar, penulis juga aktif menulis buku dan artikel ilmiah. Buku-buku dan artikel yang ditulis mencakup tema pemrograman, analisis data, dan pengelolaan proyek teknologi. Penulis pernah menjadi pembicara di berbagai konferensi teknologi nasional dan internasional. Penulis juga aktif sebagai konsultan IT dan CO-Founder LKP Unity Academy Medan. LinkedIn: www.linkedin.com/in/yusuf-ramadhan-nasution-4486912b3. Instagram: @ramadhannst

BAB 8
PERULANGAN PADA C++

Johan Suryo Prayogo
Universitas Anwar Medika, Sidoarjo
E-mail: jodimasjolie@gmail.com

PENDAHULUAN

Dalam mempelajari bahasa pemrograman, kita sering dihadapkan dengan kondisi untuk mengeksekusi serangkaian intruksi secara berulang. Proses ini sering disebut sebagai *looping* atau perulangan. Perulangan memungkinkan kita untuk menulis kode dengan lebih efisien, mengurangi penulisan intruksi yang sama berulang kali, memudahkan pembacaan dan pemeliharaan program. Perulangan adalah salah satu konsep dasar dalam pemrograman yang memungkinkan program untuk menjalankan suatu blok kode secara berulang berdasarkan kondisi tertentu. Dalam pengembangan perangkat lunak, sering kali dibutuhkan mekanisme untuk mengulang proses yang sama, baik digunakan untuk melakukan perhitungan matematika, memproses data dalam jumlah besar, atau menjalankan operasi yang bersifat repetitif. Oleh karena itu, perulangan menjadi bagian yang tak terpisahkan dalam pemrograman modern, termasuk dalam bahasa pemrograman C++.

Pada bahasa pemrograman C++ menawarkan beberapa struktur perulangan, seperti *while*, *do-while*, dan *for*, yang masing-masing memiliki karakteristik dan kegunaan spesifik. Perulangan *while* sangat tepat digunakan ketika jumlah perulangan atau iterasi tidak diketahui, sedangkan *do-while* memastikan blok kode dijalankan setidaknya satu kali sebelum mengevaluasi kondisi. Perulangan *for* sering digunakan ketika

jumlah perulangan atau iterasi sudah diketahui sebelumnya, seperti iterasi dalam *array* atau koleksi data. Keunggulan penggunaan perulangan dalam bahasa pemrograman C++ tidak hanya terletak pada efisiensi waktu dan ruang, tetapi juga pada kemampuan untuk menyederhanakan kode. Perulangan dapat membantu menghindari duplikasi kode, sehingga program menjadi lebih mudah dipahami, dipelihara, dan dioptimalkan.

Perbandingan *do-while, while* dan *for*

Aspek	*do-while*	*while*	*for*
Eksekusi awal	Selalu dieksekusi minimal satu kali	Tidak selalu diekekusi	Tidak selalu dieksekusi
Penempatan	Setelah blok kode	Sebelum blok kode	Didalam deklarasi
Cocok untuk	Validasi input	Kondisi perulangan yang tidak diketahui	Perulangan dengan batas yang jelas

WHILE

Perintah *while* adalah struktur kontrol yang memungkinkan program untuk menjalankan blok kode secara berulang selama kondisi yang ditentukan bernilai benar (*true*). Perulangan while sangat tepat digunakan ketika jumlah perulangan atau iterasi tidak diketahui sebelumnya dan tergantung pada suatu kondisi yang berubah selama program di eksekusi. Sintaks dasar dari perintah *while* dalam bahasa pemrograman C++ adalah

KELEBIHAN DAN KEKURANGAN PERULANGAN FOR

Kelebihan

1. Semua komponen (inisialisasi, kondisi, perubahan) didefinisikan dalam satu baris.
2. Cocok untuk iterasi dengan batas perulangan yang jelas.

Kekurangan

1. Kurang fleksibel dibandingkan dengan perulangan *while* untuk kondisi yang dinamis.
2. Bisa menjadi rumit untuk dibaca jika memiliki banyak logika yang kompleks.

KESIMPULAN

Perulangan adalah salah satu struktur kontrol dalam bahasa pemrograman C++ yang memungkinkan eksekusi blok kode secara berulang berdasarkan pada kondisi tertentu. Beberapa jenis perulangan utama dalam bahasa pemrograman C++ adalah *for, while* dan *do while.* Ketiga jenis perulangan ini memiliki kondisi dalam pemakaiannya masing-masing yang juga sudah dijelaskan dalam bab Perulangan ini. Untuk perulangan *For* digunakan ketika jumlah iterasi telah diketahui sebelumnya, perulangan ini efektif untuk pengulangan dengan langkah atau rentang yang sudah terdefinisi. Perulangan *While,* digunakan ketika kondisi pengulangan diperiksa sebelum eksekusi blok kode. Perulangan *While* cocok untuk situasi dimana jumlah iterasi tidak dapat ditentukan sebelumnya. Dan perulangan *Do-While* mirip dengan perulangan *While,* namun blok kode dijalankan setidaknya satu kali sebelum kondisi diperiksa.

DAFTAR PUSTAKA

Deitel, P., Deitel, H., 2017. C++ How To Program Tenth Edition, Global Edition.

Raharjo, B., 2018. Pemrograman C++ Mudah dan Cepat Menjadi Master C++, Revisi Kedua. ed. Informatika Bandung, Bandung.

Ramadhani, C., 2019. Algoritma Pemrograman dan Struktur Data. Penerbit ANDI (Anggota IKAPI), Yogyakarta.

Thareja, R., 2014. Data Structures Using C, Second Edition. ed. Oxford University Press, New Delhi.

PROFIL PENULIS

Johan Suryo Prayogo, S.Kom., M.T.

Penulis adalah seorang dosen Prodi Sistem Informasi di Universitas Anwar Medika Sidoarjo. Mengambil studi Strata 1 Teknik Informatika di Universitas Surabaya (UBAYA) serta melanjutkan Program Magister Teknik Informatika di Universitas Atma Jaya Yogyakarta (UAJY). Penulis memiliki kegemaran membaca dan mempelajari sesuatu yang baru, khususnya di bidang *UI/UX Designer*, *Web Development,* dan bahasa pemrograman. Besar harapan penulis dengan terbitnya buku ini menjadi sarana bagi mahasiswa dan Masyarakat umumnya untuk mempelajari bahasa pemrograman menggunakan C++ dengan lebih menyenangkan.

BAB 9
MANIPULASI STRING DAN OPERASINYA

Cosmas Haryawan
Universitas Teknologi Digital Indonesia, Yogyakarta
E-mail: cosmas@utdi.ac.id

PENDAHULUAN

String adalah salah satu elemen fundamental dalam pemrograman yang merepresentasikan urutan karakter, baik huruf, angka, maupun simbol, yang biasanya digunakan untuk menyimpan dan memproses teks. Dalam banyak bahasa pemrograman, *string* biasanya didefinisikan sebagai urutan karakter yang disusun secara berurutan. Meskipun aturan untuk menangani *string* berbeda-beda antara satu bahasa pemrograman dengan yang lain, konsep *string* tetap serupa, yaitu untuk mempermudah pengelolaan data berupa karakter (Busbee, 2018) Menurut Savich dan Mock (2021), dalam C++ *string* dapat dikelola melalui dua model pendekatan: **array karakter (char[])** dan **std::string**. Pendekatan ini memiliki karakteristik dan kegunaan masing-masing.

1. **Pendekatan Klasik atau disebut juga C-Style String dengan char[]**

   Biasa disingkat dengan sebutan C-Style. *String* dalam bentuk *array* karakter adalah cara tradisional yang digunakan sejak era bahasa C. *String* ini harus diakhiri dengan karakter null (\0) atau disebut *string terminating character* untuk menandai akhir string. Seperti konsep *array* pada umumnya, variabel C-Style menggunakan posisi mulai dari indeks 0 hingga sesuai panjang *string* tersebut. Untuk mengetahui panjang dari *string* tersebut, digunakan penanda sebuah karakter null di akhir *string*. Jika C++

menemukan karakter null maka pembacaan isi array selesai, dan hingga batas tersebutlah yang dianggap sebagai 1 variabel *string*. Sebagai contoh, untuk *string* "HALO", akan tersimpan di variabel sebagai berikut:

Index	0	1	2	3	4
String	H	A	L	O	\0

Savitch dan Mock (2018) menyampaikan bahwa dalam konsep C-Style terdapat istilah *partially-filled*, ini merujuk pada situasi di mana *array* karakter yang disiapkan untuk menyimpan string tidak sepenuhnya terisi oleh data string itu sendiri. *Array* tersebut memiliki kapasitas tetap yang ditentukan saat deklarasi, tetapi string yang disimpan di dalamnya mungkin hanya memanfaatkan sebagian dari kapasitas tersebut. Misal disiapkan variabel string sepanjang 8 karakter, dengan perintah str[8], kemudian diisi dengan kata HALO, maka yang terisi hanya 5 ruang di *array*, sedangkan 3 sisanya tidak terisi.

Str[0]	Str[1]	Str[2]	Str[3]	Str[4]	Str[5]	Str[6]	Str[7]
H	A	L	O	\0	?	?	?

Partially-filled ini memiliki beberapa dampak negatif, diantaranya:

- Keamanan: Bagian yang tidak terisi dalam *array* dapat mengandung nilai sampah (*garbage values*), yang dapat menyebabkan hasil yang tidak terduga jika tidak diatur dengan hati-hati.
- Pemanfaatan Memori: *Partially-filled array* bisa dianggap sebagai pemborosan memori jika kapasitas yang dialokasikan terlalu besar dibandingkan *string* aktual yang digunakan.

```
  // Mengubah menjadi huruf besar
  transform(str.begin(), str.end(), str.begin(), ::toupper);
  cout << str << endl; // Output: C++ PROGRAMMING
  // Mengubah menjadi huruf kecil
  transform(str.begin(), str.end(), str.begin(), ::tolower);
  cout << str << endl; // Output: c++ programming
  return 0;
}
```

KESIMPULAN

Manipulasi string di C++ menawarkan fleksibilitas tinggi dengan dukungan C-Style String dan kelas `*string*` modern, memungkinkan programmer untuk menangani data berbasis teks dengan efisien, fleksibel dan aman. Melalui berbagai metode yang ada, C++ menyediakan fitur yang lengkap untuk memenuhi kebutuhan pengolahan *string* dalam berbagai konteks. Hal ini akan memberikan kemudahan bagi pengembang untuk bekerja dengan data berbasis karakter, baik untuk kebutuhan sederhana maupun kompleks. Pemahaman tentang konsep dan teknik manipulasi *string* ini tidak hanya meningkatkan kemampuan pemrograman tetapi juga dapat digunakan untuk menciptakan solusi yang lebih kreatif dan optimal dalam berbagai kebutuhan sistem aplikasi.

DAFTAR PUSTAKA

Busbee, K. L., & Braunschweig, D. (2018). Programming Fundamentals A Modular Structured Approach 2nd Edition, Retrieved from https://harpercollege.pressbooks.pub/programmingfundamentals/

Savitch, W, & Mock, K. (2021). Absolute C++ 7th Edition, London:Pearson Education

Savitch, W, & Mock, K. (2018). Problem Solving With C++ Tenth Edition Global Edition, London:Pearson Education

PROFIL PENULIS

Cosmas Haryawan

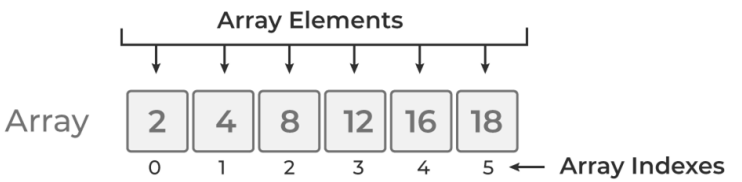
Lulus S1 Teknologi Pertanian UGM tahun 1998, kemudian Lulus S1 Teknik Informatika, Universitas Teknologi Yogyakarta (UTY) tahun 2004. Mendapatkan gelar Master of Computer Science (M.Cs) dari Ilmu Komputer MIPA UGM tahun 2014. Saat ini masih aktif sebagai dosen program studi Sistem Informasi Universitas Teknologi Digital Indonesia (UTDI) di Yogyakarta. Selain mengajar di UTDI, pernah menjadi dosen tamu juga di Fakultas Teknik Informatika (FTI) UTY dan Fakultas Ekonomi dan Bisnis (FEB) Universitas Atma Jaya Yogyakarta. Banyak mengampu mata kuliah pemrograman, dari algoritma hingga pemrograman web. Di luar kegiatan tridharma dosen, hingga saat ini masih menjadi seorang programmer dan juga IT Consultant untuk pengembangan sistem informasi perusahaan.

BAB 10
ARRAY PADA C++

Hidayatus Sibyan
Universitas Sains Al-Qur'an, Wonosobo
E-mail: hsibyan@unsiq.ac.id

PENDAHULUAN

Array merupakan salah satu struktur data dasar yang sangat penting dalam Pemrograman (Ginting et al., 2024). Array adalah struktur data yang digunakan untuk menyimpan kumpulan data dengan tipe yang sama dalam satu variabel (Mustakim et al., 2024). Elemen-elemen array disimpan dalam lokasi memori yang berurutan dan dapat diakses menggunakan indeks. Indeks array di C++ dimulai dari angka nol, sehingga elemen pertama array berada di indeks 0, elemen kedua di indeks 1, dan seterusnya (FIRLIANA & Kasih, 2018). Berikut adalah ilustrasi bagaimana elemen-elemen array tersusun secara berurutan dalam memori:



Gambar 10.1. Ilustrasi array

DEKLARASI DAN INISIALISASI ARRAY

Array merupakan struktur data penting dalam pemrograman. Untuk dapat menggunakan array, langkah pertama adalah mendeklarasikannya, dan setelah itu, menginisialisasi elemen-elemennya.

1. Sintaks Deklarasi Array

Deklarasi array di C++ digunakan untuk mendefinisikan sebuah array dengan ukuran tertentu dan tipe data elemen-elemen di dalamnya. Sintaks dasar untuk mendeklarasikan array adalah sebagai berikut (Utami et al., 2017):

```
tipe_data nama_array[ukuran];
```

Tipe_data menentukan tipe data dari elemen-elemen array (misalnya int, float, char, dll.). Nama_array merupakan nama dari variabel array. Sedangkan ukuran adalah jumlah elemen yang akan disimpan dalam array. Berikut adalah contoh deklarasi array.

```
int angka[5];     // Array untuk menyimpan 5 elemen bilangan bulat
float suhu[10];     // Array untuk menyimpan 10 elemen bilangan pecahan
char huruf[3];    // Array untuk menyimpan 3 elemen karakter
```

Dalam contoh di atas, array angka memiliki kapasitas 5 elemen, tetapi elemen-elemen tersebut belum diinisialisasi sehingga nilainya tidak terdefinisi.

2. Inisialisasi Array

Inisialisasi dilakukan pada saat deklarasi array, di mana nilai-nilai awal elemen langsung ditentukan (FIRLIANA & Kasih, 2018). Sintaks dasar inisialisasi array adalah sebagai berikut:

Output Program di atas adalah sebagai berikut.

```
Nilai siswa:
Siswa ke-1: 85 90
Siswa ke-2: 78 88
Siswa ke-3: 92 80
```

Gambar 10.7. Output program nilai

KESIMPULAN

Array adalah struktur data yang menyimpan kumpulan elemen dengan tipe data yang sama dalam memori yang berurutan. Array memberikan efisiensi dalam menyimpan dan mengakses data. Array dapat dideklarasikan dengan menentukan ukuran dan tipe datanya. Inisialisasi dapat dilakukan secara statis (nilai langsung ditentukan) atau dinamis (menggunakan perulangan untuk mengisi elemen). Array mendukung berbagai operasi seperti traversal (mengakses elemen satu per satu), pengisian nilai, pencarian elemen. Array dua dimensi dan multidimensi memungkinkan penyimpanan data dalam bentuk matriks atau tabel. Array ini digunakan untuk operasi matematika seperti penjumlahan dan transpose matriks.

DAFTAR PUSTAKA

Anita Sindar, R. M. S. (2019). *Struktur Data Dan Algoritma Dengan C++* (Vol. 1). CV. AA. RIZKY.

FIRLIANA, R., & Kasih, P. (2018). *Algoritma dan Pemrograman C++*.

Ginting, S. H. N., Effendi, H., Kumar, S., Marsisno, W., Sitanggang, Y. R. U., Anwar, K., Santiari, N. P. L., Setyowibowo, S., Sigar, T. R., & Atho'illah, I. (2024). Pengantar Struktur Data. *Penerbit Mifandi Mandiri Digital*, *1*(01).

Guntara, R. G. (2023). *ALGORITMA DAN PEMROGRAMAN DASAR: Menggunakan Bahasa Pemrograman C++ dengan*

Contoh Kasus Aplikasi untuk Bisnis dan Manajemen. CV. Ruang Tentor.

Mustakim, S., Kom, M., Susilo, I. D., Kom, M., Mokhammad Syafaat, S. T., Mukminin, A., Kom, M., Ariestiandy, D., Kom, M., & Buan, F. C. H. (2024). *STRUKTUR DATA DAN ALGORITMA*. Cendikia Mulia Mandiri.

Sussolaikah, K., Kom, S., & Kom, M. (n.d.). *Implementasi Algoritma Pemrograman Berbasis C++*. Deepublish.

Szauer, G. (2020). *Hands-On C++ Game Animation Programming: Learn modern animation techniques from theory to implementation with C++ and OpenGL*. Packt Publishing Ltd.

Utami, E., Kom, M., Dhuhita, W. M. P., Kom, S., & Kom, M. (2017). *Langkah Mudah Belajar Struktur Data Menggunakan C/C++*. Elex Media Komputindo.

PROFIL PENULIS

Hidayatus Sibyan

Lahir di Wonosobo pada tanggal 30 November 1989, adalah seorang akademisi di bidang Teknik Informatika. Menyelesaikan studi S1 Teknik Informatika di Universitas Sains Al-Qur'an (UNSIQ) Wonosobo pada tahun 2012. Mendapatkan gelar Magister Komputer (M.Kom.) pada program studi Magister Teknik Informatika STMIK Amikom Yogyakarta pada tahun 2016. Telah mengabdikan dirinya di Universitas Sains Al-Qur'an (UNSIQ) Wonosobo sebagai tenaga pendidik sejak tahun 2016 sampai saat ini. Mata kuliah yang diampu diantaranya Algoritma dan Dasar Pemrograman, Algoritma dan Struktur Data, Basis Data, Sains Data.

BAB 11
FUNGSI REKURSIF PADA C++

M. Agus Triawan
Politeknik Negeri Sriwijaya, Jurusan Teknik Komputer, Palembang
E-mail: matriawan@polsri.ac.id

PENDAHULUAN

C++ adalah bahasa pemrograman serba guna (general-purpose) yang dibangun berdasarkan bahasa C, sehingga sering ditulis dengan bahasa pemrograman C/C++ (dianggap sebagai perkembangan dari bahasa C yang mendukung pemrograman berbasis objek) (Stroustrup, 2022). Bahasa pemrograman C++ banyak mempengaruhi bahasa pemrograman lainnya yang juga populer, seperti bahasa pemrograman Java dan C# (dibaca C-Sharp) (Meyers, 2014). Sama seperti pada bahasa pemrograman lainnya, fungsi rekursif pada C++ adalah sebuah fungsi dengan konsep memanggil dirinya sendiri. Biasanya fungsi rekursif banyak digunakan pada algoritma pengurutan (sorting), serta pemrograman dinamis seperti untuk membentuk struktur *parent-child*, *tree*, atau *drop-down* seperti pada pembentukan struktur direktori dan file (Lippman et al., 2012).

Fungsi rekursif secara sederhana berfungsi untuk menyelesaikan permasalahan yang tergantung pada sub-masalah itu sendiri sebagai solusinya. Oleh karena itu, secara umum, fungsi rekrusif terdiri dari 2 (dua) bagian.

1. Basis Rekrusif (Base Case): Kondisi untuk menentukan rekursif berhenti atau untuk mencegah rekursif tanpa henti (Vandevoorde & Josuttis, 2002).
2. Langkah rekursif (Recursive Step): Kondisi atau *state* fungsi memanggil dirinya sendiri dengan argumen untuk mencapai basis rekursif (Alexandrescu, 2001).

Berikut contoh penulisan fungsi rekursif pada bahasa pemrograman C++ yang terdiri dari basis dan langkah rekursif.

```
int fungsiRekursif(int n) {
   if (n <= 1) {  // Basis rekursif
      return 1;
   } else {       // Langkah rekursif
      return n * fungsiRekursif(n - 1);
   }
}
```

Keterangan: Pada fungsi tersebut, deklarasi varible n dengan tipe integer saat dipanggil akan menjadi argumen (nilai sebenarnya) dan nilainya terus berkurang sebanyak 1 hingga n memenuhi kurang dari atau sama dengan 1 (hingga mencapai nilai basis/dasarnya).

CARA KERJA REKURSIF

Setiap kali fungsi memanggil dirinya sendiri, suatu *frame* baru (ruang atau blok-blok *main-memory* sebagai tempat program yang terdiri dari instruksi dan data disimpan sebagai *page*) akan disimpan kedalam tumpukan/*stack main-memory* yang menyimpan nilai variabel n sebagai argumen sampai basis rekursif tercapai (Shaffer, 2012).

Dapat dikatakan dari penulisan fungsi rekursif yang sederhana menjadi kelebihan fungsi rekursif itu sendiri, serta dapat diandalkan sebagai solusi alami untuk menangai masalah seperti *traversal-tree* (pohon-biner) yaitu mendatangi setiap simpul dari pohon secara sistematik tepat satu kali (Cormen et al., 2022). Namun karena saat fungsi rekursif dipanggil, sistem memori harus menyimpan status eksekusi saat ini dalam *stack*, maka akan menambah *overhead* pada *main-memory* (memori ekstra yang dikonsumsi oleh sistem untuk mengelola dan menjalankan tugas, di luar memori yang dibutuhkan untuk tugas

penggunaan fungsi rekursif yang umum, seperti dalam perhitungan faktorial (n!).

Kelebihan fungsi rekursif dapat dikatakan sebagai solusi yang elegan untuk masalah kompleks dan dapat mengurangi kode yang diperlukan. Namun kekurangannya yaitu penggunaan memori yang tinggi dan potensi untuk terjadinya *stack overflow* jika tidak diatur dengan baik (basis rekursif). Dari contoh kode fungsi rekursif sederhana untuk menghitung faktorial dan memeriksa bilangan prima, yang menunjukkan penerapan konsep rekursif dalam pemrograman. Sehingga dari pembahasan fungsi rekursif ini dapat memberikan wawasan yang berharga bagi yang ingin memahami lebih dalam tentang C++ dan penerapan fungsi rekursif, serta pentingnya pemrograman yang efisien dan efektif dalam pengembangan perangkat lunak.

DAFTAR PUSTAKA

Alexandrescu, A. (2001). *Modern C++ Design: Generic Programming and Design Patterns Applied*. Pearson Education.

Carey, J., Doshi, S., & Rajan, P. (2019). *C++ Data Structures and Algorithm Design Principles: Leverage the power of modern C++ to build robust and scalable applications*. Packt Publishing.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to Algorithms, fourth edition*. MIT Press.

Lippman, S. B., Lajoie, J., & Moo, B. E. (2012). *C++ Primer*. Pearson Education.

Meyers, S. (2014). *Effective Modern C++: 42 Specific Ways to Improve Your Use of C++11 and C++14*. O'Reilly Media.

Pai, G. A. V. (2022). *A Textbook of Data Structures and Algorithms, Volume 1: Mastering Linear Data Structures*. Wiley.

Shaffer, C. A. (2012). *Data Structures and Algorithm Analysis in C++, Third Edition*. Dover Publications.

Stroustrup, B. (2022). *A Tour of C++*. Addison-Wesley.

Vandevoorde, D., & Josuttis, N. M. (2002). *C++ Templates: The Complete Guide*. Pearson Education.

PROFIL PENULIS

M. Agus Triawan

Penulis adalah seorang pengajar di bidang teknik komputer yang berfokus pada penerapan perangkat keras dan perangkat lunak agar dapat berinteraksi dengan dunia nyata, yang dikenal sebagai *Cyber Physical System* (CPS). Ia juga merupakan pendiri platform idraya.com, sebuah tempat untuk berbagi tulisan dan artikel tentang pemrograman, sistem tertanam (embedded system), serta jaringan komputer, yang dapat menjadi sumber referensi bagi siapa saja yang ingin mempelajari teknologi komputer.

BAB 12
POINTER PADA C++

Tundo
Sekolah Tinggi Ilmu Komputer Cipta Karya Informatika, Teknik Informatika, Jakarta
E-mail: asna8mujahid@gmail.com

PENDAHULUAN

Pointer adalah salah satu fitur penting dalam C++ yang memungkinkan programmer untuk bekerja dengan alamat memori secara langsung. Meskipun konsep ini bisa tampak kompleks pada awalnya, memahami pointer dapat memberikan fleksibilitas dan efisiensi yang lebih besar dalam pemrograman. Dalam bab ini, kita akan menjelaskan apa itu pointer, bagaimana cara menggunakannya, dan mengapa pointer penting dalam pengembangan perangkat lunak.

PENGERTIAN POINTER

Pointer adalah variabel yang menyimpan alamat memori dari variabel lain. Dalam C++, setiap variabel memiliki alamat memori, dan pointer memberikan cara untuk mengakses data pada alamat tersebut.

Sintaks Dasar:

```
int a = 5;    // Variabel integer
int* p = &a;  // Pointer yang menyimpan alamat a
```

Penjelasan:

- int* p mendeklarasikan p sebagai pointer ke tipe int.
- &a adalah operator yang mengambil alamat memori dari a.

DEKLARASI DAN INISIALISASI POINTER

Pointer harus dideklarasikan dengan tipe data yang sesuai dengan variabel yang akan ditunjuk. Berikut adalah contohnya:

```
float f = 3.14;
float* ptr = &f;
```

Penjelasan:

- Pointer ptr hanya bisa menunjuk ke variabel bertipe float.

Catatan: Jika pointer tidak diinisialisasi, ia akan memiliki nilai acak. Gunakan nilai nullptr untuk menunjukkan bahwa pointer belum menunjuk ke alamat tertentu:

```
int* p = nullptr;
```

MENGAKSES DATA MELALUI POINTER

Pointer dapat digunakan untuk mengakses atau memodifikasi data variabel yang ditunjuk. Untuk melakukan ini, kita menggunakan operator dereferensi (*).

```
int x = 10;
int* p = &x;
*p = 20;  // Mengubah nilai x melalui pointer
```

Output:

- Nilai x sekarang adalah 20.

Contoh lain dari program akses data melalui Pointer

KEAMANAN POINTER

Penggunaan pointer yang tidak tepat dapat menyebabkan kesalahan seperti **segmentation fault**. Berikut adalah beberapa praktik terbaik:

1. Inisialisasi pointer dengan nullptr.
2. Jangan dereferensi pointer yang tidak valid.
3. Gunakan delete setelah alokasi dinamis.
4. Hindari kebocoran memori dengan mengelola alokasi dengan hati-hati.

KESIMPULAN

Pointer adalah elemen penting dalam C++ yang memberikan kontrol langsung terhadap memori. Meskipun kompleks, memahami pointer memungkinkan kita untuk mengimplementasikan fitur-fitur canggih, seperti manajemen memori dinamis dan struktur data tingkat lanjut. Dengan latihan, Anda dapat menguasai penggunaan pointer dan meningkatkan efisiensi serta fleksibilitas kode Anda.

DAFTAR PUSTAKA

Carey, J., Doshi, S., & Rajan, P. (2019). *C++ Data Structures and Algorithm Design Principles: Leverage the power of modern C++ to build robust and scalable applications*. Packt Publishing.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to Algorithms, fourth edition*. MIT Press.

Raharjo, B., 2018. Pemrograman C++ Mudah dan Cepat Menjadi Master C++, Revisi Kedua. ed. Informatika Bandung, Bandung.

Pai, G. A. V. (2022). *A Textbook of Data Structures and*

Algorithms, Volume 1: Mastering Linear Data Structures. Wiley.

PROFIL PENULIS

Tundo

Lahir di Pemalang pada tanggal 03 Desember 1991. Penulis menyelesaikan pendidikan Strata I pada tahun 2017 di Universitas Teknologi Yogyakarta Prodi Informatika, Magister (S2) Prodi Informatika pada tahun 2020 di Universitas Islam Negeri Sunan Kalijaga Yogyakarta. Saat ini penulis aktif mengajar di Sekolah Tinggi Ilmu Komputer Cipta Karya Informatika (STIKOM CKI) sejak tahun 2024 sebagai Dosen Tetap pada program studi Teknik Informatika. Fokus penelitian yang dilakukan adalah bidang *Machine Learning*, *Artificial Intelligent*, *Decision Support System*, *Data Mining*, dan *Fuzzy Logic*. Penulis juga aktif menjadi reviewer di berbagai Jurnal Nasional Akreditasi (SINTA 4 – SINTA 6), serta Jurnal Nasional Non Akreditasi. Selain itu penulis aktif pada Forum Kerjasama Pendidikan Tinggi (FKPT) dan beberapa organisasi ilmiah.

BAB 13
STRUKTUR PADA C++

Agung Yuliyanto Nugroho
Universitas Cendekia Mitra Indonesia
E-mail: agungyuliyanto@unicimi.ac.id

PENDAHULUAN

Struktur (struct) dalam bahasa C++ adalah salah satu fitur dasar yang sangat penting untuk pemrograman terstruktur. Struktur memungkinkan Anda untuk mendefinisikan tipe data kustom yang mengelompokkan berbagai elemen data di bawah satu nama. Ini memudahkan pengelolaan data yang saling terkait dan membantu dalam menyusun kode dengan cara yang lebih terorganisir.

1. Apa itu Struktur.
   Dalam C++, struktur (struct) adalah kumpulan variabel (disebut anggota atau fields) yang dikelompokkan bersama di bawah satu nama. Struktur memungkinkan Anda untuk membuat tipe data kustom yang dapat menyimpan data berbeda dalam satu unit.
2. Berikut adalah cara mendefinisikan dan menggunakan struktur dalam C++:

```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 13.1. Struktur dalam C++

3. 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.
4. Constructor dan Destructor dalam Struktur
   Seperti dalam kelas, Anda juga dapat mendefinisikan konstruktor dan destruktor dalam struktur untuk menginisialisasi dan membersihkan sumber daya.

dalam pemrograman C++. Dengan memahami dan memanfaatkan struktur, kamu dapat menulis kode yang lebih terstruktur, modular, dan mudah dipelihara.

DAFTAR PUSTAKA

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/

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

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

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

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

Schildt, H. (2019). C++: The complete reference (5th ed.). McGraw-Hill Education.

Stroustrup, B. (2013). The C++ programming language (4th ed.). Addison-Wesley.

Stroustrup, B. (2000). The C++ programming language (3rd ed.). Addison-Wesley.

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

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/

PROFIL PENULIS

Agung Yuliyanto Nugroho S.Kom., M.Kom., M.Par.

Ketertarikan penulis terhadap ilmu komputer dimulai pada tahun 2012 silam. Hal tersebut membuat penulis memilih untuk masuk ke Sekolah Menengah Kejuruan di SMK Negeri 1 Godean Sleman dengan memilih Jurusan Multimedia pada saat itu (MM) dan berhasil lulus pada tahun 2015. Penulis kemudian melanjutkan pendidikan ke Perguruan Tinggi dan berhasil menyelesaikan studi S1 di prodi Teknik Informatika Universitas Teknologi Yogyakarta pada tahun 2018. Dua tahun kemudian, penulis menyelesaikan studi S2 di prodi Teknik Informatika Program Pasca Sarjana Universitas Amikom Yogyakarta dan juga prodi Magister Pariwisata di Sekolah Tinggi Pariwisata Ambarrukmo Yogyakarta. Penulis memiliki kepakaran dibidang Web Technology, Data Science dan Kepariwisataan. Dan untuk mewujudkan karir sebagai dosen profesional, penulis pun aktif sebagai peneliti dibidang kepakarannya tersebut. Beberapa penelitian yang telah dilakukan didanai oleh internal perguruan tinggi dan juga Kemenristek DIKTI. Selain peneliti, penulis juga aktif menulis buku dengan harapan dapat memberikan kontribusi positif bagi bangsa dan negara yang sangat tercinta ini. Atas dedikasi dan kerja keras dalam membuat suatu karya, Republik Indonesia Kementerian Hukum Dan Hak Asasi Manusia sudah mencatat ada kurang lebih 20 karya yang sudah tercatat di surat pencatatan ciptaan sebagai salah satu kontribusi dalam melindungi hak kekayaan intelektual.

ALGORITMA DAN PEMROGRAMAN DENGAN C++

FUTURE SCIENCE
Jl. Terusan Surabaya, Gang 1 A No. 71 RT 002 RW 005, Kel. Sumbersari, Kec. Lowokwaru, Kota Malang, Provinsi Jawa Timur.
Website : www.futuresciencepress.com

IKAPI
No . 348/JTI/2022

ISBN 978-634-7037-98-5 (PDF)
9 786347 037985