**Algoritma Pemrograman C/C++**

**Pendahuluan**

Pendekatan yang digunakan di awal, siswa diajak untuk mengenali program-program kecil yang isinya adalah elemen pembangun program: ekspresi, assignment, kondisional dan loop. Pada kenyataannya, sebuah program mengandung campuran dari elemen pembangun tersebut. Siswa perlu meramu dan mengintegrasikan pengetahuan dengan "membongkar" sebuah program contoh dan mengambil elemennya untuk diadaptasi kemudian mengkonstruksi kembali sebuah program utuh untuk menyelesaikan persoalan yang dihadapinya.

Ini adalah hakekat dari Computational Thinking. Dekomposisi persoalan dilakukan dengan memecah program menjadi bagian kecil elemen sintaktikal, untuk membentuk solusi yang utuh. Pattern (pola) diharapkan dapat terbentuk mulai dari elemen dasar sintaktikal yang dipelajari melalui program kecil, menuju pola solusi untuk jenis-jenis persoalan pembangun persoalan lebih besar.

Dengan dasar pemikiran dan pendekatan tersebut, Memulai Pemrograman Dasar dengan Bahasa C/C++ ini terdiri atas beberapa modul.

* Pendahuluan: Memperkenalkan sejarah C/C++, contoh aplikasi, dan istilah-istilah yang dipakai. Juga dibahas IDE yang disarankan.
* Program Kecil dalam Bahasa C/C++: Mengenalkan bahasa C/C++ dengan program terkecil yaitu Halo Dunia.
* Inisialisasi dan Assignment: Mengenalkan apa itu variabel dan tipe dasar yang ada, seperti integer (bilangan bulat), bilangan riil (bilangan pecahan), dan karakter (huruf). Serta memahami dua cara yang dapat dilakukan untuk mengisi nilai dari sebuah variabel, yakni inisialisasi dan assignment.
* Konstanta: Mengenalkan arti konstanta, yakni variabel yang tidak dapat diubah. Serta Mengenalkan perbedaan antara konstanta dengan variabel.
* Baca-Tulis: Mengajarkan bagaimana memasukkan suatu nilai ke variabel.
* Operasi Sederhana dengan Komputer: Mengajarkan operasi-operasi yang ada di dalam C/C++, seperti aritmatika, boolean, perbandingan, dll.
* Instruksi Kondisional: Mengenalkan instruksi kondisional agar program dapat berjalan sesuai alur yang ditetapkan.
* Instruksi Perulangan: Mengenalkan instruksi perulangan agar program dapat mengulangi instruksi secara berulang, seperti for, while-do, do-while, dan for.
* Array-Tabel: Mengenalkan array untuk menyimpan banyak nilai dalam suatu bentuk matriks.
* Subprogram: Mengenalkan subprogram fungsi dan prosedur, yaitu sebuah abstraksi suatu proses komputasi yang dapat dipanggil oleh sebuah program.
* Penutup: Menguji pemahaman mengenai bahasa pemrograman C/C++ melalui 5 latihan untuk memperdalam konsep pemrograman C/C++ dalam sebuah program sederhana, seperti operasi komputasi, instruksi kondisional, instruksi perulangan, tabel atau array, serta fungsi dan prosedur.

Pembelajaran Algoritma pemrograman akan lebih banyak melakukan praktik dengan komposisi persentase kelas sebagai berikut, 20% Materi & Teori dan 80% merupakan praktik. Sehingga waktu untuk menulis kode lebih banyak daripada teori atau materi.

**Pengenalan C/C++**

C (dibaca /si:/ seperti membaca huruf C dalam bahasa inggris) adalah bahasa pemrograman gerneral-purpose dan imprative yang mendukung pemrograman terstruktur dan rekursif. Bahasa C menggunakan static typing yang artinya tipe datanya harus ditentukan secara manual. 2

General-purpose artinya bisa digunakan untuk berbagai kebutuhan. Seperti aplikasi CLI, Desktop, Sistem Operasi, IoT, Game, dan sebagainya. Bahasa C menggunakan paradigma imperative artinya penulisan kode program di bahasa C menggunakan statament atau perintah-perintah. Paradigma pemrograman adalah cara atau teknik dalam penulisan kode program. Bisa juga dibilang sebagai sudut padang dalam menyelesaikan masalah. Karena manggunakan paradigma imperative, bahasa C juga tentunya mendukung paradigma struktural dan prosedural. Namun C belum mendukung Object Oriented Programming (Pemrograman Berorientasi Object). Jika ingin menggunakan OOP, sebaiknya Anda menggunakan C++ karena di sana sudah ditambahkan fitur OOP seperti Class, Object, Interface, dan sebagainya.

Bahasa C dikembangkan oleh Dennis M. Ritchie dan Brian W. Kernighan pada awal tahun 1970. Bahasa C berkembang di lingkungan UNIX (±90% sistem operasi UNIX ditulis dalam bahasa C). Terdapat beberapa standar untuk bahasa C, standar di sini dapat diartikan sebagai guideline dalam menuliskan bahasa C. Beberapa Standar yang ada yaitu:

* Definisi Kernighan & Ritchie (K&R),
* ANSI-C (X-3.159 -1989-), [Standar yang akan Digunakan Dalam Pembelajaran]
* Definisi AT&T (untuk superset C, C++), dan
* GNU Coding Standards.

Versi pada PC misalnya:

* Lattice C,
* Microsoft C/Microsoft QuickC, dan
* Turbo C/Borland C++.

Bahasa C banyak dipakai untuk:

* Membuat sistem operasi dan program-program sistem.
* Pemrograman yang "dekat" ke perangkat keras (misalnya untuk kontrol peralatan).
* Membuat tool kit.
* Menulis program aplikasi.

Kelebihan bahasa C, sehingga terpilih untuk aplikasi-aplikasi tersebut, adalah kemampuannya untuk membuat kode yang compact, efisien tanpa mengorbankan readability (beda dengan bahasa assembly yang efisien namun susah dibaca, atau bahasa tingkat tinggi lain yang enak dibaca namun tidak efisien). Walaupun tak dapat diingkari bahwa program dalam bahasa C lebih sulit dibaca (karena compact) dibandingkan dengan bahasa tingkat tinggi yang lain.

Pada tahun 1986, dikembangkan superset C (kompatibel dengan C, namun dilengkapi dengan kemampuan pemrograman berorientasi objek) oleh Bjarne Stroustrup [Stroustrup-86], yaitu bahasa C++ (C with Class). Seperti namanya, Simbol “++” pada huruf C berarti increment dari C. Sebenarnya C++ sama seperti bahasa C, tapi memiliki fitur yang lebih banyak dibandingkan C seperti Class, Object, Interface, dan sebagainya. Karena itulah dinamakan C++ (dibaca si plus plus).

**Pengantar Dasar Algoritma Pemrograman**

Untuk memulai memahami algoritma pemrograman serta menulis sebuah kode program, kita perlu memahami dasar sebagai pengantar agar kedepannya kita dapat dengan mudah mempelajari dan mengikuti materi yang ada.

1. **Cara Menulis**

Dalam menulis program, hal pertama yang perlu diperhatikan ialah cara membaca dan menulis kode program. Sama seperti kita membaca dan menulis pada umumnya. Dalam membaca dan menulis kode program, konsep yang digunakan ialah konsep membaca dan menulis dari atas menuju kebawah. Terkadang hal tersebut disepelekan dan justru berakibat fatal dalam menulis kode program, karena akan mengakibatkan error pada program. Hal itu disebabkan oleh urutan program yang tidak sesuai dalam mengikuti kaidah/konsep penulisan dari atas kebawah. Komputer akan membaca kode program kita secara berurutan dari atas hingga kebawah ssehingga jika cara menulisnya tidak berururtan dan tidak beraturan akan mengakibatkan error yang fatal. Hal yang sederhana namun terbilang sangat penting. Itulah sebabnya diperlukan pemahaman dasar dalam menulis kode program.

1. **Pentingnya Dokumentasi**

Dalam mempelajari bahasa pemrograman kita perlu membaca dokumentasi resmi agar meminimalisir kesalahan serta error yang terjadi nantinya. Setiap bahasa pemrograman pasti memiliki standar berbeda-beda dalam penulisannya, oleh sebab itu dokumentasi sangat wajib, penting serta perlu dibaca. Anda bisa mencari dokumentasi resmi setiap bahasa pemrograman yang Anda pelajari di internet.

1. **Indentasi & Blok Program**

Indentasi merupakan memulai teks dengan spasi kosong di antara teks dan margin. Tab digunakan untuk indentasi, dan jumlah karakter dalam tab adalah 4 karakter spasi. Biasanya indentasi akan digunakan di dalam blok program.

Dalam bahasa pemrograman blok program ialah sebuah pendekatan pemrograman yang memungkinkan pengguna untuk membuat program dengan menyusun balok-balok kode yang saling berhubungan. Blok program dalam bahasa pemrograman umumnya diawali dengan tanda buka kurung kurawal “{“ serta diakhiri dengan tutup kurung kurawal “}”. Dalam beberapa kasus bahasa pemrograman lain seperti Python, kurung kurawal tidak merepresentasikan/mewakilkan keadaan blok program tetapi menggunakan simbol lain seperti “:”.

Deskripsi blok (unit dari ke ) adalah seperti yang ditunjukkan dalam contoh berikut, dan sintaksnya dijelaskan setelah indentasi ”{“ “}”.

|  |
| --- |
| for ( i = 0; i < MSG\_MAX; i++ )  { //Memasuki Blok Program  [Tab]<Code>  [Tab]<Code>  } //Mengakhiri Blok Program |

Berikut ini contoh blok program yang terdapat didalam blok program (blok program bersarang):

|  |
| --- |
| bool fs\_tf\_analizeFileUpCommand( char \*cmd, int size )  { //Memasuki Blok Program 1  [Tab]int i;  [Tab]for( i = 0; i < size; i++ )  [Tab]{ //Memasuki Blok Program 2  [Tab][Tab]<Code>  [Tab][Tab]<Code>  [Tab]} //Mengakhiri Blok Program 2  [Tab]return OK\_CMD;  } //Mengakhiri Blok Program 1 |

1. **Komentar**

Dalam menulis kode program, terkadang kita perlu menambahkan komentar dalam beberapa bagian tertentu sebagai keterangan. Mengapa demikian? Coba pikirkan jika Anda bekerja dalam tim dan tim Anda mengerjakan program yang cukup rumit dan kompleks tetapi Anda justru hanya menulis program tanpa adanya keterangan komentar. Bagaimana dengan teman tim Anda yang nantinya meneruskan pekerjaan Anda? Tentu saja akan menyulitkan tim Anda dalam meneruskan pekerjaan Anda nantinya. Itulah pentingnya sebuah komentar dalam menulis sebuah program.

Komentar dalam bahasa pemrograman dapat dibuat hanya untuk sebaris atau lebih dari satu baris. Jika hanya sebaris, umumnya komentar dapat diawali dengan “//” tanpa ada akhiran. Tetapi jika Anda ingin memasukkan komentar lebih dari satu baris, umumnya komentar diawali dengan “/\*” dan diakhiri dengan “\*/”. Dalam bahasa pemrograman lain terkadang terdapat perbedaan diantaranya python yang menggunakan awalan komentar “#” untuk komentar sebaris. Jadi jika Anda berniat mempelajari bahasa pemrograman lain harap baca kembali setiap dokumentasi bahasa pemrograman lain agar tidak salah menuliskan sintaksis komentar.

Berikut Contoh penggunaan komentar:

|  |
| --- |
| if( filePtr == null )  { //Komentar Sebaris  /\* Komentar  Lebih  Dari  Satu Baris \*/  return ERR\_CMD;  } |

Note: Komentar sebaris “//” tidak dapat digunakan dalam Bahasa C, jadi untuk membuat komentar dalam bahasa C gunakan “/\* \*/”. Komentar sebaris “//” masih dapat digunakan dalam bahasa pemrograman C++.

1. **Aturan Penamaan**

Dalam mempelajari bahasa pemrograman kita perlu memperhatikan dalam aturan penulisan nama variabel, fungsi, kelas, dsb. Aturan penamaan sebagian besar sama serta untuk mempermudah kita dalam membaca kode programnya nanti. Ada tiga hal penting yang perlu diperhatikan dalam aturan penamaan:

* Beri penamaan yang dapat membedakan setiap jenis tipe data.
* Tipe data sebagai parameter nama untuk mengetahui tipe data isi setelah dideklarasikan, selain itu akan meminimalisir terjadinya kesalahan.
* Beri penamaan yang mudah dikenali dan dipahami (jangan gunakan singkatan yang hanya diketahui oleh Anda dan Tuhan Anda seperti x = 5).

Ada hal lain yang penting untuk Anda ketahui saat membuat nama:

* Gunakan aturan penamaan ini sebagai acuan umum dan bila perlu sesuaikan aturan penamaan dengan bahasa pemrograman yang Anda gunakan, karena setiap bahasa pemrograman juga memiliki aturan penamaan yang berbeda (Baca Dokumentasi Resmi).
* Jangan menuliskan nama yang sama dengan kata kunci (if, else, for, while, dll), karena akan menimbulkan error dan tidak dapat dibaca oleh suatu program.
* Hindari penggunaan inisial seperti penggunaan satu huruf saja pada penamaan variabel/fungsi/kelas jika programnya rumit, karena akan membingungkan dan sulit dibedakan.
* Jangan mengawali nama variabel/fungsi/kelas dengan angka, karena ini akan membuat kompiler membacanya sebagai nilai, bukan variabel/fungsi/kelas. Oleh karena itu, jika variabel/fungsi/kelas kita memerlukan angka di dalamnya, maka angka tersebut tidak boleh diletakkan di awal.
* Hindari memberi nama variabel/fungsi/kelas dengan istilah umum seperti “data”. Menamakannya “data” tidak menjelaskan apa pun. Namanya akan membuat Anda bingung nantinya. Sebagai gantinya, gunakan nama yang menggambarkan nilai variabel/fungsi/kelas itu sendiri.
* Variabel dalam Bahasa Pemrograman seperti JavaScript dan lain-lain hanya boleh diawali dengan huruf atau garis bawah. Anda tidak dapat membuat variabel dengan awalan angka atau menggunakan simbol selain garis bawah.
* Nama variabel/fungsi/kelas tidak boleh mengandung spasi. Jika nama variabel/fungsi/kelas lebih dari dua kata, maka gunakan format camelCase seperti ini: NamaDepan, NamaBelakang, NamaKucing atau Anda bisa menggunakan underscore “\_” sebagai pengganti spasi jika tidak ingin menggunakan metode camelCase.
* Nama variabel/fungsi/kelas tidak boleh mengandung karakter khusus (!, /+\*= dan lain-lain).

Berikut beberapa aturan untuk membuat fungsi/kelas:

* Hindari membuat banyak parameter dalam suatu fungsi jika tidak diperlukan. Karena jika argumen terlalu banyak maka akan sulit untuk diuji atau digunakan. Jika diperlukan lebih dari tiga argumen fungsi maka dapat dikonsolidasikan dengan tim Anda. Daripada membuat banyak parameter fungsi, manfaatkan objek sebagai parameter.
* Jadikan suatu fungsi hanya melakukan satu hal karena jika fungsi yang kita buat melakukan banyak hal, maka fungsi tersebut akan sulit untuk dikompilasi dan diuji. Fungsi yang melakukan banyak hal juga tidak memiliki tujuan yang jelas. Saat membuat fungsi yang melakukan lebih dari satu hal, cobalah memecahnya. Hingga setiap fungsi benar-benar melakukan satu hal. Tentu saja, ini juga membuat kode dalam fungsi menjadi lebih bersih dan mudah dibaca.
* Nama fungsi harus mewakili tujuannya. kita juga perlu memperhatikan penamaan fungsinya. Pastikan nama yang diberikan mewakili maksud atau tugas fungsi tersebut. Hal ini juga dapat membantu pengembang lain dengan mudah mengetahui tujuan dari fungsi yang Anda buat.
* Buat fungsi untuk menghindari duplikasi kode. Jika Anda merasa sering menulis kode yang berulang-ulang. Anda harus memperhatikan kodenya. Karena kode berulang adalah kandidat kuat untuk membuat suatu fungsi. Tujuannya tak lain agar kode tersebut bisa digunakan kembali, hanya dengan memanggil fungsinya saja.

1. **Running Program**

Untuk menjalankan kode program kita perlu mencari tahu terlebih dahulu apakah bahasa pemrograman tersebut dijalankan menggunakan metode compiler atau interpreter.

* Compiler: Compiler menerjemahkan kode dari bahasa pemrograman tingkat tinggi ke dalam kode mesin sebelum program dijalankan. Compiler akan membaca keseluruhan file dan akan menjalankan keseluruhannya serta mengeluarkan hasilnya (Berjalan dengan baik atau menghasilkan error) meskipun terdapat code yang error didalam code programnya. Contoh Bahasa Pemrograman yang menggun akan compiler ialah C/C++, dan Jawa.
* Interpreter: Interpreter menerjemahkan kode yang ditulis dalam bahasa pemrograman tingkat tinggi ke dalam kode mesin baris demi baris saat kode dijalankan. Interpreter akan membaca baris perbaris lalu jika terjadi error program akan langsung berhenti tepat di baris yang mengalami error tersebut. Contoh bahasa pemrograman yang menggunakan interpreter ialah Python.

1. **Type of Error**

Belum lengkap kalo tidak membahas error dalam algoritma pemrograman. Error merupakan sebuah kesalahan yang terjadi dalam penulisan kode program. Umumnya error terdapat banyak macam jenisnya tergantung setiap masing-masing bahasa pemrograman. Berikut ini beberapa tipe error yang berkemungkinan akan terjadi kedepannya ketika mempelajari bahasa pemrograman:

* Logical error atau kesalahan logika merupakan error yang paling sulit untuk dideteksi. Hal itu disebabkan error yang satu ini terjadi bukan karena adanya kesalahan penulisan atau kesalahan pada proses runtime. Namun, juga karena adanya kesalahan dari programmer dalam penggunaan algoritma. Saat logikanya salah, pasti output yang dihasilkan juga akan ikut salah. Kemudian untuk mendeteksinya cukup sulit dan memakan waktu. Sebagian besar dari logical error terjadi karena adanya kesalahan dalam perhitungan atau menggunakan variabel yang salah. Saat terjadi logical error biasanya tidak akan membuat program berhenti secara total. Pasalnya, program akan tetap bisa berjalan normal, tapi tidak bisa berfungsi seperti yang diharapkan.
* Selain logical error, salah satu jenis error pada programming yang paling sering terjadi adalah syntax errors atau kesalahan tata bahasa. Jenis error yang satu ini disebut paling banyak terjadi dalam programming. Meski begitu, mendeteksinya ternyata cukup mudah dan tidak serumit seperti logical error. Syntax error bisa terjadi saat ada kesalahan ejaan pada salah satu kata kunci atau saat ada kesalahan dengan struktur kodenya. Ketika terjadi syntax error maka kode tidak akan berjalan dengan normal karena komputer tidak dapat memahaminya. Untuk memperbaiki jenis error yang satu ini hanya membutuhkan ketelitian untuk mencari adanya kesalahan penulisannya.
* Runtime error adalah kesalahan yang terjadi saat sedang menjalankan suatu program. Ada beberapa penyebab terjadinya jenis error yang satu ini. Misalnya, kesalahan dalam proses input, kesalahan perhitungan, dan proses output. Saat terjadi runtime error, potensi terjadinya crash pada program menjadi cukup besar. Jika ingin memperbaiki jenis error yang satu ini, programmer mau tidak mau harus kembali lagi ke fase coding untuk mencari kesalahannya.
* Compilation error terjadi saat proses di mana program yang ditulis dalam bahasa tingkat tinggi dikonversi ke bentuk yang dapat dibaca oleh mesin. Dalam proses tersebut dapat muncul beberapa jenis error seperti syntax error. Bahkan, terkadang meskipun kode syntax sudah benar, tapi compilation error masih bisa tetap terjadi. Hal itu dikarenakan adanya masalah di compiler itu sendiri. Namun, jangan khawatir karena jenis error ini bisa diperbaiki pada fase development.
* Interfacing error kemungkinan besar dapat terjadi karena adanya ketidaksesuaian program software dengan interface perangkat keras yang digunakan. Sementara itu, pada kasus aplikasi web biasanya terjadi karena penggunaan protokol web yang salah.
* Arithmetic error adalah salah satu jenis error pada programming yang tergolong logical error. Namun, kesalahan ini melibatkan perhitungan matematika, bukan pada penulisan bahasa pemrograman. Hal ini umumnya terjadi karena komputer tidak bisa mengerjakan perhitungan yang dimasukkan olehmu. Sebagai contoh, ketika kamu meminta komputer untuk melakukan pembagian dengan angka 0 (nol). Karena tidak bisa dilakukan secara matematis, hal tersebut dapat menyebabkan error yang mencegah program untuk berjalan semestinya. Untuk mengatasinya, kamu bisa melakukan hal-hal seperti:
  + memahami operasi matematika dasar
  + menghindari kesalahan dalam menggunakan tanda kurung atau urutan operasi
  + menggunakan angka negatif dalam melakukan perhitungan
* Resource error adalah jenis error pada programming ketika nilai variabel yang kamu masukkan terlalu besar sehingga membuatnya melebihi batas maksimal pemrograman. Sebuah program akan menggunakan sejumlah sumber daya yang disediakan komputer untuk membuatnya bisa berjalan. Ketika program membutuhkan sumber daya lebih banyak dari yang bisa disediakan komputer, hal ini akan menyebabkan resource error. Untuk mengatasinya, kamu bisa menggunakan aplikasi load-testing untuk mengetahui apa yang akan terjadi ketika kamu menjalankan program yang sama secara bersamaan.
* Semantic error adalah jenis error pada programming yang serupa dengan logical error. Meski begitu, perbedaannya adalah logical error menghasilkan data yang salah. Sementara, semantic error menghasilkan suatu hal yang tidak memiliki makna sama sekali.

Terjadinya kesalahan atau error di atas sebenarnya bisa dihindari jika kamu lebih teliti dan melakukan perencanaan yang baik saat di tahap menulis code program.

**SILABUS ALGORITMA & PEMROGRAMAN C/C++**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Kompetensi Dasar** | **Materi Pokok dan Uraian Materi** | **Kegiatan Pembelajaran** | **Penilaian** | | **Alokasi Waktu** | **Alat** |
| **Teknik Tes** | **Hasil Tes** |
| Memahami tentang Algoritma Pemrograman Dasar, Sejarah Bahasa C/C++, serta Kegunaan Bahasa Pemrograman C/C++ | Pengenalan tentang Sejarah serta kegunaan Bahasa Pemrograman C/C++ serta Mempelajari Pseudocode dan Diagram Alur Sebelum menulis Kode Program, serta pembahasan terkait computational thinking. | * Materi serta Tanya Jawab terkait sejarah dan kegunaan bahasa pemrograman C/C++ * Praktik Membuat Diagram Alur * Praktik Membuat Pseudocode | Tes Tulis | * Diagram Alur * Isian (Pseudocode) | * 50 Menit (Materi) * 10 Menit (Tanya Jawab) * 30 Menit (Praktik Diagram Alur) * 30 (Praktik Pseudocode) | Draw.io |
| Melakukan Instalasi IDE Membuat, Program Pertama serta Struktur Dasar Pemrograman C/C++ | Pengenalan tentang Integrated Development Environment Dev-C++ serta struktur dasar dari kode sumber C/C++. | * Melakukan Instalasi IDE Dev-C++ * Praktik membuat Program dasar * Penjelasan Struktur Kode Program Dalam C/C++ | Tes Tulis | * Kode Sumber Program Pertama | * 60 Menit (Instalasi) * 20 Menit (Program Pertama) * 40 Menit (Penjelasan Struktur Program) | Dev-C++ |
| Fungsi Input serta Output, Variabel, Konstanta, Tipe Data dan Operator | Pengenalan Fungsi Input scanf(), getline(), cin serta fungsi output printf(), dan cout. Pengenalan tipe data dasar String, Int, float, dsb, Pengenalan Variabel beserta jenis-jenisnya dan konstan serta pengenalan beberapa operator penugasan, operator logika, operator pembanding, operator bitwise, operator aritmatika, dsb. | * Materi serta Tanya Jawab * Praktikum dari apa yang dipelajari | Tes Tulis | * Kode Sumber Program | * 30 Menit (Materi) * 90 Menit (Praktikum) | Dev-C++ |
| Struktur Percabangan | Pengenalan struktur percabangan serta keyword if, else if, dan else, dan switch, case, break, serta penggunaan operator logika dan operator pembanding pada Struktur Percabangan. Dan terakhir serta mempelajari Percabangan bersarang. | * Materi serta Tanya Jawab * Praktikum dari apa yang dipelajari | Tes Tulis | * Kode Sumber Program | * 30 Menit (Materi) * 90 Menit (Praktikum) | Dev-C++ |
| Struktur Perulangan I | Pengenalan For, While, dan Do While dan tujuan kegunaan masing-masing perulangan. | * Materi serta Tanya Jawab * Praktikum dari apa yang dipelajari | Tes Tulis | * Kode Sumber Program | * 30 Menit (Materi) * 90 Menit (Praktikum) | Dev-C++ |
| Struktur Perulangan II | Perulangan Bersarang serta kombinasi perulangan dan percabangan. | * Materi serta Tanya Jawab * Praktikum dari apa yang dipelajari | Tes Tulis | * Kode Sumber Program | * 30 Menit (Materi) * 90 Menit (Praktikum) | Dev-C++ |
| Struktur Data Array & List | Pengenalan terhadap struktur data Array & List serta melakukan praktik pengelolaan tipe data array dengan menggunakan perulangan | * Materi serta Tanya Jawab * Praktikum dari apa yang dipelajari | Tes Tulis | * Kode Sumber Program | * 30 Menit (Materi) * 90 Menit (Praktikum) | Dev-C++ |
| Prosedur & Fungsi I | Pengenalan terhadap Fungsi beserta jenis-jenisnya serta kegunaannya, parameter/argument, dan nilai kembalian | * Materi serta Tanya Jawab * Praktikum dari apa yang dipelajari | Tes Tulis | * Kode Sumber Program | * 30 Menit (Materi) * 90 Menit (Praktikum) | Dev-C++ |
| Prosedur & Fungsi II | Pengenalan terhadap fungsi rekursif, dan membuat program yang kompleks menggunakan fungsi | * Materi serta Tanya Jawab * Praktikum dari apa yang dipelajari | Tes Tulis | * Kode Sumber Program | * 30 Menit (Materi) * 90 Menit (Praktikum) | Dev-C++ |
| Tipe Data Enum, Structure & Union | Pengenalan terhadap tipe data Enum, Struct dan Union serta kegunaannya | * Materi serta Tanya Jawab * Praktikum dari apa yang dipelajari | Tes Tulis | * Kode Sumber Program | * 30 Menit (Materi) * 90 Menit (Praktikum) | Dev-C++ |
| Pointer | Pengenalan terhadap konsep tipe data pointer, address, dan nilai pada Bahasa Pemrograman C/C++ | * Materi serta Tanya Jawab * Praktikum dari apa yang dipelajari | Tes Tulis | * Kode Sumber Program | * 30 Menit (Materi) * 90 Menit (Praktikum) | Dev-C++ |
| Ujian | Ujian Praktikum untuk memecahkan program yang diajari serta pembahasan penyelesaian soal di akhir. | * Tes Kemampuan Dari Hasil Pembelajaran | Tes Tulis | * Kode Sumber Program | * 100 Menit (Ujian) * 20 Menit (Pembahasan) | Dev-C++ |