



BUKU AJAR

MEMBUKA DUNIA KODE

Belajar Dasar-Dasar dari Pemograman

Disusun Oleh:

Dhavi Novrizal
Rifqi Wahid Dhoiri
Hafiz Nur Fauzan
Fahmi Putra
Erva Zoya Indriyani

Diterbitkan Oleh:



**UNIVERSITAS
TEKNOKRAT
INDONESIA**

KATA PENGANTAR

Dengan segala rasa syukur dan puji bagi Tuhan Yang Maha Esa, kami haturkan penghargaan setinggi-tingginya atas karunia-Nya yang tak terhingga. Modul ini kami persembahkan sebagai bentuk rasa syukur atas nikmat sekaligus sebagai upaya kami dalam membagikan pengetahuan dan wawasan yang berharga.

Di tengah keanekaragaman ilmu pengetahuan dan informasi, modul ini dirancang untuk menjadi jembatan pengetahuan, membimbing pembaca melalui suatu perjalanan yang penuh makna. Kami berharap modul ini dapat memberikan kontribusi positif bagi pengembangan pengetahuan dan pemahaman Anda terhadap Dunia pemrograman

Tentu saja, pembuatan modul ini tidak terlepas dari kerjasama dan dukungan berbagai pihak.

Kami Mengucapkan Terima kasih kepada:

1. Miss lili yang telah menjadi pembimbing kami saat pembuatan modul
2. Kepala Sekolah SMA YP UNILA Yang telah mengizinkan kami untuk bisa menyalurkan modul yang kami buat ke sekolah yang dipimpin Beliau

Dan Kami ingin menyampaikan terima kasih kepada semua yang telah terlibat dalam proses ini, baik itu melalui kontribusi tulisan, saran-saran berharga, maupun dukungan moral. Keberhasilan modul ini tidak terlepas dari upaya bersama.

Semoga modul ini dapat menjadi panduan yang bermanfaat dan membantu Anda dalam menjelajahi dan memahami lebih dalam tentang Pemrograman Dasar. Teruslah menjaga semangat belajar, karena pengetahuan adalah kunci untuk membuka pintu-pintu keberhasilan.

Akhirnya, kami berdoa semoga modul ini dapat memberikan manfaat yang nyata dan memberikan inspirasi baru dalam setiap langkah perjalanan belajar Anda.

Semoga Tuhan Yang Maha Esa senantiasa memberkati perjalanan pengetahuan kita semua.

Bandar Lampung,..

Penyusun

UNIVERSITAS TEKNOKRAT INDONESIA

KAMPUSNYA SANG JUARA

DAFTAR ISI

COVER.....	i
KATA PENGANTAR	ii
DAFTAR ISI	iii
DAFTAR TABEL	v
DAFTAR GAMBAR	vi
BAB 1	1
PENDAHULUAN.....	1
1.1 Latar Belakang.....	1
1.2 Sejarah Pemrograman	2
1.3 Tujuan Pembelajaran	3
1.4 Definisi Algoritma.....	5
1.5 Struktur Algoritma	6
1.6 Kegiatan Belajar Algoritma Menggunakan Bahasa Natural	9
BAB 2	23
DEV C++	23
2.1 Pengertian Dev C++	23
2.2 FITUR DEV C++	24
2.3 Menu File	27
2.4 Menu Edit.....	28
2.5 Menu Execute	30
2.6 Menu Search	31
BAB 3	33
JENIS-JENIS OPERATOR DALAM C++	33
3.1 Pengertian Operator Dalam C++	33
3.2 Operator Aritmatika	33
3.3 Operator Penugasan	36
3.4 Operator Pembandingan	38
3.5 Operator Logika.....	40
3.6 Operator Lainnya.....	42
BAB 4	46
PERINTAH DASAR PEMROGRAMAN C++	46
4.1 Pendahuluan	46

UNIVERSITAS TEKNOKRAT INDONESIA

KAMPUSNYA SANG JUARA

4.2	Integer, Float Dan Double	49
4.3	String	51
4.4	Boolean, If, If Else Dan If Else If.....	53
4.5	For, While, Do While Dan Case Of	56
4.6	Array.....	59
BAB 5	61
MENGANALISIS PROSES PADA PROGRAM PARKIR.....		61
5.1	Bagian [1]	65
5.2	Bagian [2]	66
5.3	Bagian [3]	67
5.4	Bagian [4]	67
5.5	Bagian [5]	70
5.6	Bagian [6]	76
5.7	Bagian [7]	78
BAB 6	79
KESIMPULAN.....		79
6.1	Kesimpulan	79
6.2	Ucapan Terima Kasih Dan Penutup	80

DAFTAR TABEL

Tabel 1.1 Perintah Pada Pseudocode.....	13
Tabel 1.2 Simbol simbol pada flowchart.....	18
Tabel 3.1 Operator aritmatika dalam C++	34
Tabel 3.2 Operator Penugasan.....	37
Tabel 3.3 Operator Pembandingan.....	38
Tabel 3.4 Operator Logika.....	40
Tabel 3.5 Operator Lain	42

DAFTAR GAMBAR

<i>Gambar 1.1 Buku Aljabar</i>	<i>5</i>
<i>Gambar 1.2 Visualisasi sederhana tipe data dan variabel</i>	<i>14</i>
<i>Gambar 1.3 Flowchart Menyalakan Komputer</i>	<i>20</i>
<i>Gambar 2.1 Aplikasi DEV C++.....</i>	<i>23</i>
<i>Gambar 2.2 Menu File</i>	<i>27</i>
<i>Gambar 2.3 Menu Edit.....</i>	<i>28</i>
<i>Gambar 2.4 Menu Tools</i>	<i>30</i>
<i>Gambar 2.5 Menu Search</i>	<i>31</i>
<i>Gambar 3.1 Contoh Kode Aritmatika A.1</i>	<i>34</i>
<i>Gambar 3.2 Contoh hasil Kode Aritmatika A.2</i>	<i>34</i>
<i>Gambar 3.3 Contoh Semua Operator Aritmatika A.3</i>	<i>35</i>
<i>Gambar 3.4 Contoh Hasil Kode A.4</i>	<i>35</i>
<i>Gambar 3.5 Contoh Kode Aritmatika A.5</i>	<i>36</i>
<i>Gambar 3.6 Contoh Hasil Kode Aritmatika A.6</i>	<i>36</i>
<i>Gambar 3.7 Contoh Kode penugasan B.1</i>	<i>37</i>
<i>Gambar 3.8 Contoh Hasil Kode penugasan B.2</i>	<i>37</i>
<i>Gambar 3.9 Contoh Kode Pembandingan C.1</i>	<i>39</i>
<i>Gambar 3.10 Contoh Kode Pembandingan C.2</i>	<i>39</i>
<i>Gambar 3.11 Contoh Kode Logika D.1</i>	<i>41</i>
<i>Gambar 3.12 Contoh Kode Ternary E.1</i>	<i>42</i>
<i>Gambar 3.13 Contoh Kode Ternary E.2</i>	<i>43</i>
<i>Gambar 3.13 Contoh Kode Ternary E.3</i>	<i>43</i>
<i>Gambar 3.14 Contoh Kode Increment dan decrement</i>	<i>44</i>
<i>Gambar 3.15 Contoh Hasil Kode increment dan decrement</i>	<i>44</i>
<i>Gambar 4.1 Program Sesuai Cerita Diatas</i>	<i>47</i>
<i>Gambar 4.2 Program Yang menggunakan Integer</i>	<i>50</i>
<i>Gambar 4.3 Program Yang Menggunakan Float</i>	<i>50</i>
<i>Gambar 4.4 Program Yang Menggunakan Double</i>	<i>50</i>
<i>Gambar 4.5 Program Yang Menggunakan const/Konstanta</i>	<i>51</i>
<i>Gambar 4.6 Program Yang Menggunakan String.....</i>	<i>52</i>
<i>Gambar 4.7 Program Yang Menggunakan boolean</i>	<i>54</i>
<i>Gambar 4.8 Program Yang Menggunakan If</i>	<i>54</i>
<i>Gambar 4.9 Program Yang Menggunakan If Else</i>	<i>54</i>
<i>Gambar 4.10 Program Yang Menggunakan If else If.....</i>	<i>55</i>

UNIVERSITAS TEKNOKRAT INDONESIA

KAMPUSNYA SANG JUARA

<i>Gambar 4.11 Program Yang Menggunakan For</i>	<i>57</i>
<i>Gambar 4.12 Program Yang Menggunakan While</i>	<i>57</i>
<i>Gambar 4.13 Program Yang Menggunakan Do While</i>	<i>58</i>
<i>Gambar 4.14 Program Yang Menggunakan Case Of.....</i>	<i>58</i>
<i>Gambar 4.15 Program Yang Menggunakan Array</i>	<i>60</i>
<i>Gambar 5.1 Bagian 1.....</i>	<i>61</i>
<i>Gambar 5.2 Bagian 2.....</i>	<i>61</i>
<i>Gambar 5.3 Bagian 3.....</i>	<i>61</i>
<i>Gambar 5.4 Bagian 4.....</i>	<i>62</i>
<i>Gambar 5.5 Bagian 5.1.....</i>	<i>62</i>
<i>Gambar 5.6 Bagian 5.2.....</i>	<i>63</i>
<i>Gambar 5.7 Bagian 6.....</i>	<i>63</i>
<i>Gambar 5.8 Bagian 7.....</i>	<i>64</i>
<i>Gambar 5.9 Output 1</i>	<i>67</i>
<i>Gambar 5.10 Output 2</i>	<i>68</i>
<i>Gambar 5.11 Output 3</i>	<i>69</i>
<i>Gambar 5.12 Output 4</i>	<i>70</i>
<i>Gambar 5.13 Output 5</i>	<i>72</i>
<i>Gambar 5.14 Output 6</i>	<i>73</i>
<i>Gambar 5.15 Output 7</i>	<i>75</i>
<i>Gambar 5.16 Output 8</i>	<i>77</i>
<i>Gambar 5.17 Output 9</i>	<i>78</i>

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Saat ini perkembangan teknologi mengalami kemajuan yang sangat pesat khususnya pada bidang teknologi informasi dan komunikasi. Kebutuhan akan akses dan pengolahan informasi yang cepat sudah menjadi kebutuhan masyarakat modern. Misalnya, hampir semua layanan yang saat ini disediakan oleh pemerintah, swasta, dunia usaha, korporasi, dan lain-lain sudah memanfaatkan sistem informasi, baik berupa program aplikasi komputer (computer application), internet, atau sejenisnya. Meskipun program komputer ini penting bagi penyedia layanan, program ini juga sangat berguna bagi masyarakat umum sebagai konsumen karena membuat layanan menjadi lebih cepat dan mudah.

Anda dapat membayangkan apa jadinya jika supermarket tidak memiliki aplikasi pembayaran. Setiap barang harus diberi harga dan kasir harus menggunakan kalkulator untuk menghitung semua pembelian. Jika seorang pelanggan membeli keranjang belanjaan yang penuh dengan barang, berapa lama waktu yang dibutuhkan untuk melayani pembelinya?. Tentu saja hal ini memakan banyak tenaga dan waktu. Aplikasi mesin kasir yang dilengkapi pembaca barcode membuat segalanya menjadi cepat dan mudah. Pelanggan tidak perlu menunggu lama atau mengantri.

Demikian pula, layanan di tempat lain, seperti bank, rumah sakit, dan kantor pajak, sebagian besar terintegrasi ke dalam sistem informasi dalam bentuk komputer aplikasi.

- Pernahkah Anda memikirkan bagaimana program komputer dibuat?
- Berapa lama waktu yang dibutuhkan untuk membuatnya?

Aplikasi komputer ini dibuat dengan menggunakan perangkat lunak pengembangan aplikasi seperti Java, Visual Basic, dan Delphi.

Dalam membuat suatu aplikasi, seorang pengembang perangkat lunak harus terlebih dahulu mengidentifikasi masalah dan tujuan aplikasi tersebut, kemudian mengembangkan solusi masalah tersebut dalam bentuk langkah-langkah yang disebut algoritma, kemudian mengubahnya menjadi kode pemrograman sesuai dengan teknologinya. Ada yang digunakan.

terlepas dari Java, VB, Delphi, dll.

Oleh karena itu, untuk mengembangkan aplikasi dan perangkat lunak, Anda perlu mengetahui konsep dasar algoritma dan pemrograman.

1.2 Sejarah Pemrograman

Terciptanya bahasa pemrograman beriringan dengan sejarah mesin dan komputer. Awal mula bahasa pemrograman dimulai dari Antikythera yang berasal dari Yunani kuno. Antikythera adalah kalkulator yang menggunakan beberapa tuas dan konfigurasi untuk menjalankannya. Pada tahun 1200an Ismail Al-Jazari, seorang ilmuwan pada masa kejayaan Islam membangun sebuah mesin bernama Automata, sebuah robot burung merak yang bergerak dengan menggunakan hydropower (aliran air).

Cikal bakal bahasa pemrograman pertama kali muncul pada tahun 1822, sebuah mesin bernama Difference Engine diciptakan oleh Charles Babbage, mahasiswa di universitas Cambridge Inggris. Namun mesin buatan Babbage hanya bisa mengeluarkan satu jenis output. Barulah 10 tahun kemudian Charles Babbage mengembangkan mesin pengolah data itu hingga mencapai versi kedua tahun 1849. Perjuangan Babbage diteruskan oleh anaknya, Henry Prevost. Prevost membuat kopian dari perhitungan algoritma mesinnya dan mengirim ke berbagai institusi di dunia.

Dengan tersebarnya algoritma mesin Prevost, perkembangan semakin terjadi. Di tahun 1854, George Boole menemukan sistem logika yang disebut logika Boole. Logika ini menyatakan hubungan hubungan lebih besar, lebih kecil, sama dengan dan tidak sama dengan.

Pengembangan logika ini terus berkembang dari tahun ke tahun hingga seorang ilmuwan Jerman bernama Konrad Zuse membuat sebuah mesin kalkulator biner dengan nama Z-1 pada tahun 1935. Kemudian pada tahun 1939, Zuse dipanggil untuk mengabdikan pada militer dengan membuat Z-2 dan dilanjutkan dengan Z-3 dan Z-4.

Ketika sedang mengembangkan Z-4, Zuse sadar bahwa bahasa pemrograman dengan bahasa mesin terlalu rumit. Bahasa mesin ini tergolong bahasa tingkat rendah, karena hanya kumpulan kode 0 dan 1, atau ya dan tidak. Setelah melakukan penelitian selama setahun, pada tahun 1945 barulah tercipta bahasa pemrograman tingkat tinggi

pertama didunia, yaitu Plankalkul (Plan Kalkulus). Dengan Plankalkul terbukti bisa menciptakan mesin catur komputer pertama didunia.

Short Code dicetuskan pada tahun 1949 sebagai bahasa pemrograman tingkat tinggi pertama untuk mengembangkan komputer elektronik yang diciptakan oleh John Mauchly. Namun programnya harus ditranslasikan ke dalam bahasa mesin setiap dijalankan, ini membuat kinerja program dalam memproses kode memakan waktu yang cukup lama.

Alick Glennie dari Universitas Manchester mengembangkan bahasa pemrograman Autocode di awal tahun 1950an. Sebagai bahasa pemrograman, bahasa ini menggunakan kompiler yang mengkonversi secara otomatis bahasanya ke bahasa mesin. Awal bahasa pemrograman pertama digunakan pada tahun 1952 untuk komputer Mark 1 di Universitas Manchester.

John W. Backus membuat proposal ke atasannya di IBM (International Business Machines Corporation) untuk mengembangkan sebuah bahasa alternatif yang lebih praktis dari bahasa assembly untuk memprogram IBM 704 mainframe computer dengan nama Formula Translation atau yang kita kenal dengan FORTRAN. Kompiler FORTRAN berhasil diselesaikan pada April 1957.

Selanjutnya ada FLOW-MATIC yang dibuat oleh Grace Hopper. FLOW-MATIC resmi dipublikasikan pada tahun 1959 dan membawa pengaruh besar untuk pembuatan bahasa pemrograman COBOL (Common Business Oriented Language), bahasa pemrograman yang pada tahun 1959 banyak digunakan pada mainframe dan komputer mini.

Perkembangan bahasa pemrograman semakin pesat dengan adanya bahasa C. Dennis Ritchie dan Brian Kernighan menciptakan C awalnya untuk mesin DEC PDP-11. Dengan adanya bahasa C banyak bahasa baru bermunculan seperti C++, Java, C#, dan banyak lagi. Bahasa pemrograman akan terus berevolusi menjadi semakin mudah digunakan seiring dengan berkembangnya jaman.

1.3 Tujuan Pembelajaran

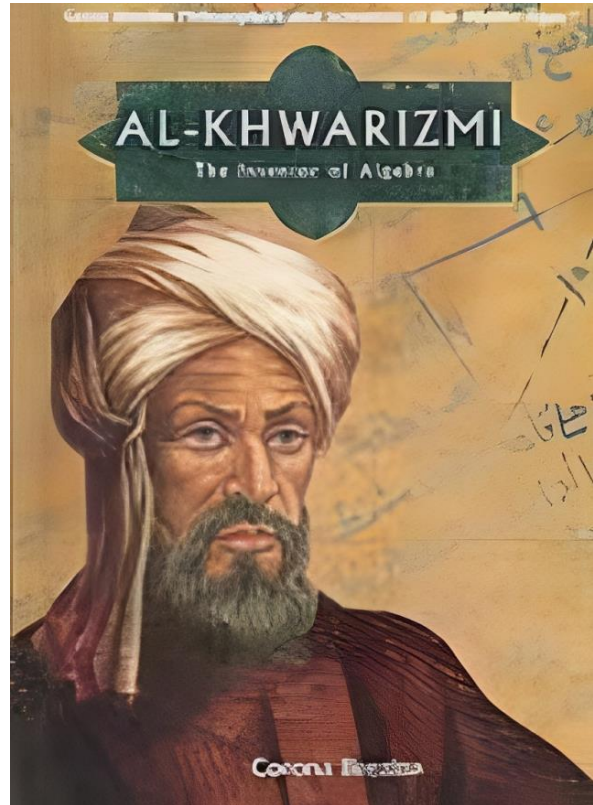
Tujuan Akhir atau output dari modul ini adalah membuat program parkir, memberikan pemahaman yang kokoh tentang dasar-dasar pemrograman dan kemudian menerapkannya dalam menciptakan solusi yang efisien untuk manajemen parkir. Melalui pemahaman konsep-konsep tersebut, pembaca diharapkan mampu merancang

dan mengembangkan program komputer yang dapat mengatasi permasalahan parkir dengan menggunakan keterampilan pemrograman yang telah dipelajari.

Dengan demikian, buku ini akan membawa Anda dalam perjalanan dari pemahaman dasar-dasar pemrograman hingga aplikasinya dalam menciptakan solusi inovatif dalam pengelolaan parkir yang efisien.

1.4 Definisi Algoritma

Definisi suatu algoritma Dilihat dari istilah algoritma, Anda bisa nama seorang matematikawan muslim yang bernama Abu Ja'far Muhammad Ibnu Musa Al-Khuwarizmi (780 M-850) yang oleh orang Barat disebut AlKhuwarizm sebagai algoritma yang ditafsirkan proses menghitung dengan angka arab. Salah satu karya monumentalnya adalah buku disebut kitab Al Jabar Wal-Muqabala yang artinya "Kitab Pemulihan dan Reduksi" (Kitab pemulihan dan pengurangan) yang menjadi pionir istilah "aljabar" digunakan. sampai sekarang. Seiring berjalannya waktu, istilah "algoritma" telah berubah menjadi suatu algoritma yang kemudian diinterpretasikan sebagai metode perhitungan (calculation method).



Gambar 1.1 Buku Aljabar

umum Dalam bahasa Indonesia kata algoritma diserap oleh algoritma. Secara definisi, algoritma adalah serangkaian langkah untuk memecahkan suatu masalah disusun secara sistematis dan logis. Dalam beberapa konteks, suatu algoritma dapat diartikan sebagai suatu urutan langkah-langkah tertentu (tertentu) untuk menyelesaikan pekerjaan. Jadi algoritmanya tidak melakukan itu diartikan sebatas perhitungan komputer saja, namun dapat diartikan lebih luas dalam kehidupan sehari-hari. Resep masakan merupakan contoh algoritma yang ada dalam kehidupan sehari-hari Petunjuk Pemasangan AC, Petunjuk Pemasangan Komputer,

Petunjuk Pemasangan Perangkat Lunak, panduan pengisian tanda listrik, grafik aktivitas merupakan bentuk berbeda dari algoritma yang ada dalam kehidupan sehari-hari.

Dalam pengertian pertama, algoritma adalah tahapan penyelesaian yang logis masalah, yaitu langkah-langkah algoritma harus logis, sesuai dengan tujuannya

tercapai dan nilai kebenarannya dapat ditentukan. Misalnya saja dalam sebuah resep memasak, ada tahap “Masak bahan x 30 menit” atau merakit komputer, "sambungkan kabel VGA dari monitor ke komputer." Kedua langkah ini adalah langkah logis, karena kebenarannya sudah diketahui dengan jelas. Kalau A cukup direbus 15 menit saja sudah bisa dikatakan bahwa langkah-langkah tersebut tidak tepat atau salah, atau mis. kabel VGA kemudian tidak tersambung jelas bahwa langkah-langkah tersebut tidak tepat. Langkah logis juga bisa diartikan tidak ambigu. langkah "tambahkan garam. Cukup” merupakan contoh langkah atau proses yang ambigu atau mempunyai makna ganda. Mencicipi bisa berarti setengah sendok teh, satu sendok teh atau satu sendok makan, tidak ada kriteria tertentu, maka antara satu pengguna dengan satu pengguna memberikan hasil yang sama tidak sama, dalam hal ini tampaknya. Demikian pula misalnya ada perintah-perintah dalam suatu algoritma “Tambahkan x ke bilangan tersebut” juga ambigu karena bilangannya seperti itu Harus dijumlahkan apakah bilangan itu asli atau nyata, genap atau ganjil, tidak ada nilai yang tetap.

Selain itu, penyusunan langkah-langkahnya harus sistematis atau terstruktur dengan baik aturan/sistem tertentu sesuai dengan tujuan yang dicapai. Artinya masuk Algoritma harus diikuti atau diamati dengan baik untuk mencapai hasil akhir diinginkan Oleh karena itu, suatu algoritma biasanya memiliki urutan yang ditetapkan untuk setiap langkah Pengguna algoritma dapat mengikuti semua instruksi yang diberikan dengan benar. Selain itu Langkah-langkah algoritma juga harus terbatas, artinya berhenti setelah eksekusi beberapa langkah. Karena tujuan suatu algoritma adalah untuk menemukan solusi. Jika Algoritma ini memiliki jumlah langkah yang tidak terbatas, yang berarti ia mencoba mencari solusi itu tidak berhasil. Jadi dapat dikatakan algoritma tersebut tidak berguna atau tidak berguna.

1.5 Struktur Algoritma

Struktur algoritma Inti dari algoritma ini adalah mencari solusi dari permasalahan tersebut. menyelesaikan masalah, algoritma meminta spesifikasi input sesuai kebutuhan, mengolahnya melalui beberapa tahapan dan memberikan hasilnya sebagai solusi masalah

Secara umum struktur Algoritma terdiri dari 3 bagian, yaitu::

1. **Nama/judul** Algoritma Nama Algoritma memberikan gambaran secara singkat apa tujuan dari Algoritma, misalkan nama resep masakan, petunjuk melakukan sesuatu, jadwal kegiatan, langkah-langkah penyelesaian sebuah masalah, dan sebagainya. Pemberian nama Algoritma disarankan singkat dan jelas, namun sudah mewakili maksud dari algoritma. Dalam Algoritma komputer biasanya nama algoritma dituliskan tanpa menggunakan spasi, misalkan Algoritma VolumeBalok atau Algoritma Volume_Balok
2. **Bagian Deklarasi** Bagian deklarasi merupakan tahap persiapan dari algoritma. Pada bagian ini dijelaskan kebutuhan agar algoritma dapat berjalan. Istilah lainnya di sinilah alat dan bahan didefinisikan. Dalam algoritma pemrograman, bagian deklarasi menjelaskan input (masukan) apa saja yang akan diproses oleh algoritma termasuk jenis data input (tipe data), juga output apa yang akan dihasilkan serta semua hal yang akan dipakai dalam algoritma. Yang didefinisikan dalam algoritma ini termasuk variabel, tipe data, konstanta, nama prosedur, tipe, dan fungsi,
3. **Bagian Deskripsi** Pada bagian ini dijelaskan serangkaian langkah-langkah (instruksi) atau pernyataan (statement) untuk memproses alat dan bahan atau inputan untuk menghasilkan output sesuai yang diharapkan. Langkah-langkah dalam algoritma dituliskan dari atas ke bawah. Urutan penulisan menentukan urutan perintah

Berikut ini gambaran struktur sebuah Algoritma;

Algoritma NAMA_ALGORITMA { Penjelasan mengenai algoritma, yang berisi uraian singkat mengenai apa yang dilakukan oleh algoritma }

DEKLARASI { Semua bahan dan alat yang dibutuhkan selama algoritma dijalankan, atau dalam algoritma meliputi variabel yang dipakai, nama tipe, konstanta, nama prosedur dan nama fungsi didefinisikan di sini }

DESKRIPSI : { Semua langkah, proses, atau statement algoritma dituliskan di sini, biasanya langkah-langkah diberikan penomoran untuk memudahkan penelusuran dan organisasi) }

Berikut adalah contoh algoritma sederhana untuk membuat donat:

RESEP MEMBUAT DONAT }

Judul Algoritma

Siapkan bahan-bahan:

- 500 gram tepung terigu
- 200 ml susu hangat
- 100 gram gula pasir
- 10 gram ragi instan
- 2 butir telur
- 50 gram margarin
- Sejumput garam

Bagian Deklarasi

1. Campurkan tepung terigu, gula pasir, dan ragi instan dalam sebuah mangkuk besar.
2. Buat lubang di tengah campuran tepung, tuangkan susu hangat dan tambahkan telur.
3. Aduk perlahan sambil mencampurkan tepung dari pinggiran lubang.
4. Tambahkan margarin dan garam ke dalam campuran dan uleni hingga adonan kalis dan elastis.
5. Tutup adonan dengan kain bersih dan diamkan selama 1 jam hingga adonan mengembang.
6. Setelah mengembang, bagi adonan menjadi beberapa bagian kecil.
7. Bulatkan masing-masing bagian menjadi bola kecil dan letakkan di atas loyang yang sudah dialasi kertas roti.
8. Diamkan lagi selama 30 menit hingga donat mengembang.
9. Panaskan minyak dalam wajan dengan api sedang.
10. Goreng donat hingga kedua sisi berwarna kuning keemasan.
11. Angkat donat dan tiriskan dari minyak.
12. Taburkan gula halus atau gula glas di atas donat yang masih hangat.
13. Donat siap disajikan.
14. Selesai

Bagian Deskripsi

1.6 Kegiatan Belajar Algoritma Menggunakan Bahasa Natural

Salah satu cara mempresentasikan algoritma adalah dengan bahasa natural atau bahasa sehari-hari. Algoritma dengan bahasa natural atau bahasa sehari dapat dengan mudah kita jumpai di kehidupan sehari-hari, kali ini, kita akan mempelajari bagaimana menyajikan algoritma dalam bahasa sehari-hari.

Secara garis besar, algoritma dapat disajikan dengan 3 cara, yaitu:

1. Menggunakan bahasa natural (bahasa sehari-hari)
2. Menggunakan Pseudocode
3. Menggunakan Flowchart

Mari kita bahas 3 hal tersebut, seperti berikut:

1. Penggunaan Bahasa Natural

Penggunaan Bahasa Natural dalam algoritma sering dijumpai pada algoritma di kehidupan sehari-hari. Bahasa yang digunakan tentunya bahasa yang dipahami oleh pengguna algoritma. Misalkan saja petunjuk perawatan sepeda motor, tentu saja akan disediakan dengan bahasa di mana motor tersebut dijual. Jika Motor tersebut dijual di Indonesia tentu saja akan disediakan petunjuk dalam bahasa Indonesia, jika dijual di beberapa negara dengan bahasa yang berbeda-beda biasanya petunjuk akan disajikan dalam banyak bahasa (multi lingual). Karena algoritma yang baik tentu algoritma yang dapat dipahami dan dijalankan oleh penggunanya.

Penyajian algoritma dalam bahasa natural, menggunakan kalimat deskriptif, yaitu menjelaskan secara detail suatu algoritma dengan bahasa atau kata-kata yang mudah dipahami. Penyajian algoritma ini cocok untuk algoritma yang singkat namun sulit untuk algoritma yang besar. Selain itu algoritma ini akan sulit dikonversi ke bahasa Pemrograman.

Penyajian Algoritma yang Baik Menurut Donald Ervin Knuth, yang dikenal dengan Bapak "Analisis Algoritma", algoritma yang baik dan benar harus memiliki kriteria-kriteria berikut ini:

- Input
- Output
- Finite
- Definite

- Efisien
- 1. **Input** Algoritma memiliki nol input atau lebih dari pengguna. Setiap algoritma pasti memiliki input. Yang dimaksud dengan nol input dari pengguna adalah bahwa algoritma tidak mendapatkan masukan dari pengguna, tapi semua data inputan yang digunakan algoritma tidak dari pengguna secara langsung, namun semua data yang akan diproses sudah dideklarasikan oleh algoritma terlebih dahulu. Sebagai contoh sebuah algoritma menghitung 100 bilangan genap yang pertama tidak memerlukan input dari pengguna karena sudah diketahui bahwa banyaknya bilangan genap adalah 100. Berbeda jika algoritma tersebut digunakan untuk menghitung n bilangan genap pertama, dengan nilai n dari pengguna. Berarti ada masukan dari pengguna yaitu n .
- 2. **Output** Algoritma minimal harus memiliki 1 output. Tujuan dari algoritma adalah memberikan penyelesaian dari suatu permasalahan dengan langkah-langkah tertentu. Penyelesaian itulah output dari algoritma yang dimaksud. Output dapat berupa apa saja, teks, file, video, suara, dan lain-lain atau suatu nilai yang disimpan untuk digunakan algoritma lain atau disimpan di basis data.
- 3. **Finite** (Terbatas) Algoritma yang baik haruslah mempunyai langkah-langkah terbatas, yang berakhir pada suatu titik di mana algoritma itu akan berhenti dan menghasilkan suatu output. Algoritma tidak boleh berjalan terus –menerus tanpa titik henti, hingga menyebabkan hang atau not responding jika diterapkan pada komputer. Ketika sebuah algoritma berjalan terus menerus (infinite), maka ini mengindikasikan ada kesalahan yang dibuat oleh programmer dalam mengembangkan algoritma.
- 4. **Define** (Pasti) Makna dari langkah logis pada definisi algoritma terdahulu tercermin dari langkah-langkah yang pasti, tidak ambigu atau bermakna ganda. Suatu program harus mempunyai arah dan tujuan yang jelas, kapan mulai dan kapan berakhir. Dalam menyusun langkah-langkah dalam algoritma perlu dihindari kata-kata seperti secukupnya, beberapa, sesuatu, sebentar, lama, atau kata lain yang tidak terukur dengan pasti. Pemberian nomor pada algoritma dapat membantu pengguna mengikuti setiap langkah

dengan pasti hingga mencapai akhir dari algoritma, yaitu solusi dari permasalahan.

5. **Efisien** Program menghasilkan output yang benar, itu wajib. Tapi bagaimana jika output yang benar itu dilakukan dengan waktu yang lama padahal ada algoritma lain yang lebih cepat? Hal ini menunjukkan bahwa setiap algoritma, khususnya jika sudah diterapkan pada pemrograman, mempunyai waktu eksekusi (running time). Algoritma disebut efisien jika untuk mendapatkan suatu solusi tidak memerlukan memori yang banyak, proses yang berbelit-belit dan tidak perlu. Jika algoritma terlalu banyak melakukan hal-hal yang tidak perlu akan menyebabkan waktu eksekusi menjadi lebih lama.

2. Menggunakan Pseudocode

Istilah **Pseudocode** terdiri dari dua gabungan kata, yaitu kata pseudo yang berarti semu dan kata code yang berarti kode. Pseudocode atau kode semu dapat diartikan sebagai deskripsi dari algoritma pemrograman yang dituliskan secara sederhana dibandingkan dengan sintaksis bahasa pemrograman. Tujuannya, agar lebih mudah dibaca dan dipahami manusia.

Ia bukanlah sebuah bahasa pemrograman, karena sebuah bahasa pemrograman harus memiliki aturan dalam penulisan kodenya. Sementara pseudocode sendiri tidak memiliki aturan yang spesifik atau baku dalam penulisannya, karena itu ia tidak dikategorikan sebagai bahasa pemrograman.

Kamu dapat membuat pseudocode ini sebelum mulai menulis sintaks dengan bahasa pemrograman. Hal tersebut bertujuan agar kamu mendapatkan gambaran bagaimana alur dari program yang akan dibuat.

➤ Fungsi Pseudocode

Selain agar lebih mudah untuk dipahami oleh manusia khususnya oleh programmer, pseudocode juga memiliki fungsi yang lain. Berikut adalah fungsinya:

- Dapat digunakan sebagai alat untuk dokumentasi.
- Untuk mempermudah proses penerjemahan menjadi suatu bahasa pemrograman
- Dapat digunakan untuk proses mencari sebuah ide tanpa harus memikirkan implementasi dari suatu bahasa pemrograman khusus.

- Lebih mudah mengembangkan aplikasi yang dibuat.

➤ **Struktur Pada Pseudocode**

Pada umumnya sebuah pseudocode memiliki tiga bagian penyusun. Bagian-bagian itu terdiri dari:

- **Judul**

Sesuai dengan namanya, bagian ini digunakan untuk menunjukkan judul dari algoritma yang akan ditulis oleh programmer

- **Deklarasi**

Deklarasi berisi keterangan seperti variabel atau konstanta yang digunakan pada algoritma

- **Deskripsi**

Deskripsi berisi yaitu ialah proses dari algoritma, ia dapat diartikan sebagai inti dari pseudocode, kamu dapat menulis segala proses pada bagian ini contohnya seperti proses kondisional (if/else) , perulangan (for), atau operasional (penjumlahan, pengurangan, dan lainnya)

Contoh:

algoritma penjumlahan_bilangan

Deklarasi

A,b : integer {variabel input}

Hasil : integer {Variabel output}

Deskripsi

Read(a,b)

Hasil \leftarrow a+b

Write(hasil)

➤ **Notasi pseudocode**

Setelah mengetahui pengertian dan fungsi dari pseudocode, kamu juga harus mengetahui notasi apa saja yang digunakan untuk mengetahui proses yang terjadi. Berikut ini adalah beberapa notasinya.

- **INPUT**

Input digunakan untuk menunjukkan proses memasukkan suatu isi variabel

- **OUTPUT**

Output digunakan untuk menunjukkan proses keluar setelah proses terjadi

- **WHILE**

While digunakan untuk sebuah perulangan yang memiliki iterasi awal

- **FOR**

For digunakan untuk sebuah penghitungan iterasi

- **REPEAT – UNTIL**

Digunakan untuk sebuah perulangan yang memiliki kondisi akhir

- **IF – THEN – ELSE**

Digunakan untuk mengambil sebuah keputusan dari beberapa kondisi.

➤ **Penulisan/Perintah pada pseudocode**

Beberapa Perintah yang ada pada pseudocode

Pernyataan	Algoritma/Penulisan pada pseudocode	Perintah pada Program C++
Input/variabel yang dimasukkan	Read(a)	cin>>a
Output/hasil atau juga keluaran dari sebuah proses	Write(a)	Cout<<a
Sama dengan	←	=
Penjumlahan	+	+
Pengurangan	-	-
Perkalian	X	*
Pembagian	div	/
Sisa bagi / modulus, yaitu sisa bagi dari pembagian, Contoh: 4 / 2 = maka modulusnya/sisa baginya adalah 0, 4 / 3 = maka modulusnya adalah 1	Mod	%
Increment i/ Penambahan “nilai” dari suatu variabel	To	++
Decrement i/ pengurangan dari “nilai” dari suatu variabel	downto	--

Tabel 1.1 Perintah Pada Pseudocode

Dan setelah kita mengenal apa itu pseudocode, kita harus mengerti dahulu apa itu variabel dan juga tipe data agar kita dapat lebih memahami contoh yang akan diberi setelah kita belajar apa itu variabel maupun tipe data, yaitu sebagai berikut:

A. PENGENALAN VARIABEL

Variabel adalah suatu konsep dalam pemrograman yang digunakan untuk menyimpan dan merujuk pada nilai tertentu di dalam sebuah program komputer. Variabel memiliki nama yang unik yang digunakan untuk mengidentifikasinya dalam lingkup program. Ketika Anda membuat variabel dalam sebuah program, Anda memberikan nama pada suatu lokasi di memori komputer yang akan digunakan untuk menyimpan nilai-nilai atau informasi tertentu.

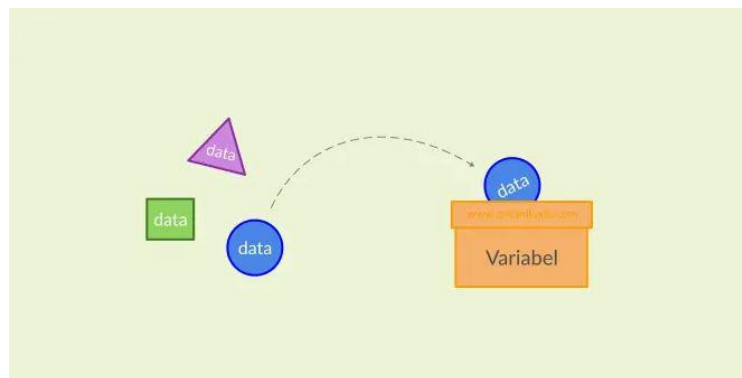
Seperti dalam matematika, Kita pasti sering menemukan x dan y .

Dan dalam pemrograman itu adalah hal yang sama. x dan y tersebut memiliki tugas yang sama seperti di matematika, yaitu menyimpan nilai

Contohnya ialah $x=3$; dan $y=2$;

Masih kurang mengerti?

Anggap saja **variabel** ialah wadah dan **tipe data** adalah jenis benda yang akan disimpan dalam benda tersebut, Seperti gambar berikut:



Gambar 1.2 Visualisasi sederhana tipe data dan variabel

Karakteristik Variabel:

1. **Nama Unik:** Setiap variabel harus memiliki nama yang unik di dalam lingkup programnya. Nama variabel ini digunakan untuk mengidentifikasi lokasi memori yang menyimpan nilai tertentu

Contoh: Variabel_A, Variabel_B, V_1, V_2, C, X, Y

2. **Tipe Data:** Variabel memiliki tipe data yang menentukan jenis nilai yang dapat disimpan di dalamnya, seperti bilangan bulat, bilangan desimal, karakter, atau tipe data lainnya.

Contoh: Variabel_A : Integer (Tipe data Integer)

3. **Lokasi Memori:** Variabel merepresentasikan lokasi spesifik dalam memori komputer yang digunakan untuk menyimpan nilai-nilai tertentu.
4. **Nilai:** Variabel dapat menyimpan nilai yang dapat diubah selama jalannya program, sesuai dengan jenis tipe data yang ditentukan. Nilai dalam variabel bisa diubah, dimanipulasi, atau diakses selama program berjalan

Dan Berikut adalah aturan aturan yang harus diikuti ketika kita membuat suatu variabel dalam pemrograman.

Aturan Dalam Membuat Variabel:

1. **Tidak Menggunakan Spasi:** Sebuah variabel tidak boleh menggunakan spasi pada pseudocode ataupun dalam pemrograman langsung
Contoh Yang Salah: Variabel A
2. **Harus Selalu Konsisten Dalam Penulisan Variabel:** Ketika kita membuat variabel di deklarasi maka variabel tersebut harus sama bentuknya pada deskripsi
Contoh Yang Salah:
Deklarasi
Variabel_A
Deskripsi
variabel_a
3. **Tidak Diawali Dengan Angka:** Dalam membuat sebuah variabel, awalan dari sebuah variabel tidak boleh menggunakan angka
Contoh Yang Salah: 1A,1Bilangan

Variabel adalah konsep penting dalam pemrograman yang memungkinkan programmer untuk menyimpan, mengelola, dan memanipulasi data dalam suatu program. Setiap bahasa pemrograman memiliki aturan dan sintaksis yang berbeda

dalam penggunaan variabel, tetapi konsep dasarnya tetap sama: menyimpan nilai yang dapat diubah dalam suatu lokasi di dalam memori komputer.

B. PENGENALAN TIPE DATA

Dalam pemrograman, terdapat beberapa tipe data sederhana yang digunakan untuk menyimpan nilai tunggal atau spesifik. Setiap tipe data memiliki karakteristiknya sendiri dan digunakan untuk menyimpan jenis data tertentu. Berikut penjelasan yang lebih rinci mengenai masing-masing tipe data sederhana yang disebutkan dalam teks Anda:

1. Bilangan Integer

Tipe data ini digunakan untuk menyimpan bilangan bulat, termasuk bilangan bulat negatif, nol, dan bilangan bulat positif. Misalnya: 1, 488, -22, 0, 456, dan lainnya. Beberapa bahasa pemrograman membagi tipe data integer menjadi kategori berbeda sesuai dengan rentang nilainya. Contohnya, dalam Pascal, terdapat byte, shortint, integer, Word, dan longint dengan rentang nilai yang berbeda.

2 Bilangan Real

Tipe data ini digunakan untuk menyimpan bilangan pecahan atau irasional yang disajikan dalam bentuk koma. Contohnya: 1.5, 458.543, -0.569, 22/7, Phi (π), $\sqrt{2}$, dan lainnya. Tipe data ini berbeda dari bilangan integer karena dapat menyimpan bilangan pecahan atau non-bulat.

3 Karakter

Data karakter merupakan tipe data yang berisi satu karakter tunggal, seperti huruf, digit, atau simbol. Tipe data ini biasanya diapit oleh tanda petik. Contohnya: 'a', 'H', '6', '^', '>'. Meskipun ada angka 6, karena diapit oleh tanda petik, nilai tersebut dianggap sebagai karakter bukan sebagai bilangan.

4 String

Tipe data string adalah susunan dari satu atau lebih karakter. Biasanya digunakan untuk menyimpan teks atau kata-kata. Misalnya: "abc", "HAPPY", "100102001"."SayaHebat!" Dalam beberapa bahasa pemrograman, untuk menggunakan tipe data string, perlu menentukan batasan maksimal panjang string yang bisa disimpan dalam variabel.

5 Boolean

Tipe data ini hanya memiliki dua nilai, yaitu Benar atau Salah, yang merepresentasikan keadaan logika True(Benar) atau False(Salah). Digunakan untuk menyimpan status kebenaran suatu pernyataan. Contohnya: statusKelulusan (Benar atau Salah), StatusPrima, StatusMenikah (Benar atau Salah).

Masing-masing tipe data memiliki penggunaan yang spesifik tergantung pada jenis data yang ingin disimpan. Penggunaan tipe data yang tepat membantu efisiensi dalam penggunaan memori dan mempermudah manipulasi data dalam program.

C. MENGENAL FLOWCHART **APA ITU FLOWCHART?**

Diagram alir atau flowchart adalah jenis diagram dengan simbol-simbol tertentu yang menggambarkan urutan proses secara mendetail dan hubungan antar prosesnya. Flowchart sering digunakan di berbagai bidang, baik bisnis, teknologi, bahkan dalam kehidupan sehari-hari. Manfaat utama flowchart adalah kemampuannya untuk menggambarkan berbagai langkah dalam suatu proses supaya lebih jelas dan mudah dipahami.

Dalam diagram alir, setiap langkah diwakili oleh simbol yang memiliki arti tertentu. Simbol-simbol ini membantu menjelaskan tugas atau keputusan yang harus diambil pada setiap langkah serta menjelaskan kemungkinan risiko dan hasilnya.

Flowchart juga dapat menggunakan anak panah atau garis penghubung untuk menunjukkan urutan dan aliran langkah-langkah yang harus diikuti.

Dengan menggunakan simbol-simbol tersebut, flowchart akan memudahkan Anda memahami suatu proses secara keseluruhan dan memberikan panduan yang jelas dalam pengambilan keputusan.

D. FUNGSI FLOWCHART

Setelah memahami pengertian flowchart, kini saatnya Anda mempelajari berbagai fungsinya. Fungsi utama flowchart adalah untuk memberikan gambaran dari sebuah proses yang kompleks.

Namun, fungsi flow chart tak berhenti di situ saja lho. Berikut ini beberapa fungsi pembuatan diagram alir yang perlu Anda ketahui:

- Menyajikan informasi yang jelas dan ringkas.
- Memberikan gambaran proses yang rumit agar lebih mudah dipahami.

UNIVERSITAS TEKNOKRAT INDONESIA

KAMPUSNYA SANG JUARA

- Membantu dalam pengambilan keputusan dengan menggambarkan berbagai kemungkinan yang terjadi.
- Memudahkan analisis seperti identifikasi risiko dan langkah-langkah yang menjadi hambatan.

➤ Macam-Macam Simbol Flowchart

	Flow Direction symbol Yaitu simbol yang digunakan untuk menghubungkan antara simbol yang satu dengan simbol yang lain. Simbol ini disebut juga connecting line.		Simbol Manual Input Simbol untuk pemasukan data secara manual on-line keyboard
	Terminator Symbol Yaitu simbol untuk permulaan (start) atau akhir (stop) dari suatu kegiatan		Simbol Preparation Simbol untuk mempersiapkan penyimpanan yang akan digunakan sebagai tempat pengolahan di dalam storage.
	Connector Symbol Yaitu simbol untuk keluar - masuk atau penyambungan proses dalam lembar / halaman yang sama.		Simbol Predefine Proses Simbol untuk pelaksanaan suatu bagian (sub-program)/prosedure
	Connector Symbol Yaitu simbol untuk keluar - masuk atau penyambungan proses pada lembar / halaman yang berbeda.		Simbol Display Simbol yang menyatakan peralatan output yang digunakan yaitu layar, plotter, printer dan sebagainya.
	Processing Symbol Simbol yang menunjukkan pengolahan yang dilakukan oleh komputer		Simbol disk and On-line Storage Simbol yang menyatakan input yang berasal dari disk atau disimpan ke disk.
	Simbol Manual Operation Simbol yang menunjukkan pengolahan yang tidak dilakukan oleh computer		Simbol magnetik tape Unit Simbol yang menyatakan input berasal dari pita magnetik atau output disimpan ke pita magnetik.
	Simbol Decision Simbol pemilihan proses berdasarkan kondisi yang ada.		Simbol Punch Card Simbol yang menyatakan bahwa input berasal dari kartu atau output ditulis ke kartu
	Simbol Input-Output Simbol yang menyatakan proses input dan output tanpa tergantung dengan jenis peralatannya		Simbol Dokumen Simbol yang menyatakan input berasal dari dokumen dalam bentuk kertas atau output dicetak ke kertas.

Tabel 1.2 Simbol simbol pada flowchart

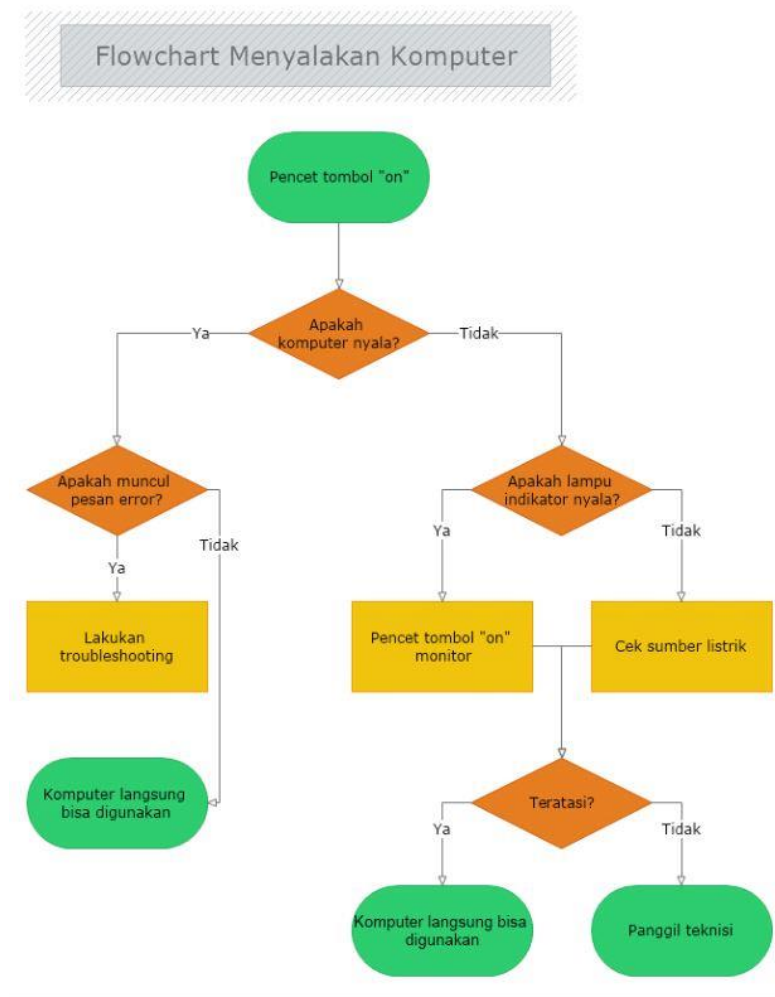
Karena dalam penyusunan flowchart adalah mengandalkan simbol pada grafisnya, maka ada banyak jenis simbol yang digunakan untuk flowchart. Simbol tersebut menunjukkan urutan suatu prosedur, mulai dari input, pemrosesan, hingga output.

[Simbol flowchart](#) yang berbeda memiliki arti konvensional yang berbeda. Arti dari beberapa simbol yang lebih umum adalah sebagai berikut:

1. Terminal: Simbol terminal merepresentasi titik awal ataupun akhir pada sistem.

2. Process: Sebuah simbol berbentuk kotak yang menunjukkan beberapa operasi tertentu.
3. Dokumen: Simbol ini mewakili proses yang dicetak, seperti dokumen atau laporan.
4. Decision: Simbol decision merepresentasi keputusan atau titik percabangan. Garis yang keluar dari bentuk simbol ini menunjukkan kemungkinan situasi yang berbeda, yang mengarah ke sub-proses yang berbeda.
5. Input / Output: Mewakili proses keluar masuk suatu proses tanpa tergantung pada peralatan yang ada.
6. On-Page Reference: Simbol ini berfungsi untuk menyambungkan aliran proses keluar masuk di halaman yang sama.
7. Off-Page Reference: Simbol ini berfungsi untuk menyambungkan aliran proses keluar masuk di halaman yang berbeda.
8. Delay: Mengidentifikasi penundaan atau hambatan.
9. Flow: Simbol berupa garis atau anak panah untuk menghubungkan satu simbol flowchart dengan simbol lain agar saling terhubung membentuk proses.

Berikut Adalah
Contoh Flowchart
Menyalakan
Komputer :



Gambar 1.3 Flowchart Menyalakan Komputer

➤ Jenis-jenis flowchart

1. System Flowchart

System flowchart adalah contoh flowchart yang menunjukkan suatu alur kerja dalam bentuk bagan. Flowchart sistem bisa untuk mengetahui apa saja yang sedang dikerjakan oleh sistem secara keseluruhan disertai penjelasan mengenai prosedurnya.

Penggambaran flowchart sistem yang berupa proses dan data bisa dilakukan dengan offline tanpa terhubung dengan komputer ataupun online dengan terhubung pada komputer.

Contoh flowchart sederhana dari system flowchart offline yaitu mesin tik dan kalkulator. Flowchart sistem juga bisa digunakan untuk mempresentasikan sistem organisasi, pemerintahan, perusahaan, pabrik, dan lembaga.

2. Document Flowchart

Document Flowchart memiliki istilah lain seperti form flowchart dan flowchart paperwork. Jadi, jika Anda mendengar salah satu dari istilah tersebut, itu sama saja merujuk pada flowchart dokumen.

Document flowchart memiliki fungsi untuk meneliti alur form dan laporan dari bagian tertentu ke bagian lain, ini bisa berupa alur form, pencatatan, pemrosesan, hingga penyimpanan laporan.

Sederhananya, contoh flowchart ini yaitu penggambaran suatu manajemen pelaporan dari organisasi, perusahaan, atau lembaga lain.

3. Schematic Flowchart

Schematic flowchart memiliki kemiripan dengan flowchart sistem, menunjukkan gambaran sistem atau prosedur.

Meskipun begitu, flowchart ini menggunakan simbol-simbol lebih kompleks dibandingkan flowchart sistem atau flowchart lainnya. Misalnya peripheral, gambar-gambar yang memiliki kaitan dengan komputer, atau alat lain yang biasa dipakai pada sistem.

Fungsi schematic flowchart adalah sebagai alat komunikasi antara analis sistem dengan orang lain yang asing pada simbol-simbol flowchart pada umumnya untuk menghemat waktu menjelaskan simbol abstrak dan meminimalisir kemungkinan kesalahpahaman. Sehingga pembaca bisa memahaminya dengan mudah melalui gambar-gambar yang familiar.

4. Program Flowchart

Contoh flowchart selanjutnya adalah contoh [flowchart](#) program yang identik dengan proses pembuatan program atau aplikasi. Langkah pembuatan program akan dijelaskan dengan rinci melalui bagan flowchart yang disusun oleh programmer. Flowchart ini dapat membantu pembaca memahami langkah pembuatan program dengan urutan yang tepat.

Flowchart ini digunakan oleh programmer untuk mempresentasikan tahapan instruksi pada program komputer, kemudian analis sistem akan menggunakannya untuk mempresentasikan urutan langkah pada prosedur atau operasi sistem.

5. Process Flowchart

Flowchart proses adalah suatu teknik visualisasi rekayasa dari industrial untuk menganalisis urutan langkah-langkah suatu proses pada prosedur tertentu.

Contoh flowchart proses bisa ditemukan di bidang manufaktur, ini karena flowchart proses digunakan sebagai perekayasa industrial untuk mempelajari suatu proses dan mengembangkannya.

BAB 2
DEV C++



Gambar 2.1 Aplikasi DEV C++

2.1 Pengertian Dev C++

Setelah kita mengenal beberapa dasar dalam pemrograman, seperti yang paling dasar yaitu Algoritma, Setelah itu kita belajar bagaimana cara ‘menerjemahkan’ bahasa komputer tersebut agar menjadi bisa terbaca oleh orang biasa menggunakan bahasa natural, pseudocode dan flowchart maka kita akan lanjut materi belajar kita.

Kali ini kita akan belajar alat/aplikasi yang mengizinkan kita agar bisa membuat program sebenarnya, sebelum kita belajar bagaimana caranya, kita akan mempelajari alatnya, yaitu Dev C++

Dev C++ adalah lingkungan pengembangan (IDE - *Integrated Development Environment*) untuk bahasa pemrograman C++. Dev C++ dikembangkan oleh Bloodshed Software dan merupakan IDE yang populer di kalangan pemrogram C++ karena sederhana, ringan, dan mudah digunakan. Serta Dev C++ Bersifat open source yang artinya yaitu gratis, bisa digunakan oleh berbagai kalangan, karena itulah mengapa Dev C++ adalah salah satu compiler yang paling diminati oleh berbagai programmer didunia ini

Dev C++ (Development Environment for C++) dikembangkan oleh Colin Laplace, yang bekerja di Bloodshed Software. Berikut adalah sejarah singkatnya:

1. Versi Awal:

Versi awal Dev C++ adalah versi 4.0.0.0, yang dirilis pada tahun 1998. Sejak itu, beberapa pembaruan dan versi iteratif telah dirilis, dengan versi terakhir dari Bloodshed Software adalah Dev C++ 5.11, yang dirilis pada tahun 2005.

2. Penggunaan Compiler GCC:

Dev C++ menggunakan GCC (GNU Compiler Collection) sebagai compiler-nya, memberikan akses ke compiler yang kuat dan open source.

3. Populer di Kalangan Pemula:

Dev C++ menjadi populer di kalangan pemula dan pengembang C++ yang mencari lingkungan pengembangan yang ringan dan mudah digunakan.

4. Kesuksesan dan Penerimaan Komunitas:

IDE ini mendapatkan penerimaan yang baik dari komunitas pengembang C++ dan menjadi salah satu pilihan utama untuk belajar dan mengembangkan aplikasi C++.

5. Proyek Berhenti Pengembangan:

Meskipun Dev C++ meraih kesuksesan, pengembangan resmi IDE ini berhenti. Versi terakhir yang dirilis oleh Bloodshed Software adalah Dev C++ 5.11 pada tahun 2005.

6. Penerus dan Pengembangan Lanjutan:

Meskipun pengembangan resmi Dev C++ berhenti, beberapa proyek turunan dan pengembangan lanjutan oleh komunitas pengembang dapat ditemukan secara online.

7. Alternatif Modern:

Dengan berakhirnya dukungan resmi, beberapa alternatif modern seperti Code::Blocks, Visual Studio Code, dan Qt Creator telah menjadi pilihan yang lebih populer di kalangan pengembang C++.

Sejarah Dev C++ mencerminkan keberhasilannya sebagai IDE C++ ringan pada masanya, tetapi pengembangan yang berhenti telah mendorong pengguna untuk beralih ke solusi pengembangan yang lebih modern dan aktif.

2.2 FITUR DEV C++

Fitur Dev C++ meliputi:

1. Integrated Development Environment (IDE):

- Editor Kode Sumber: Dev C++ menyertakan editor kode sumber yang memfasilitasi penulisan dan pengeditan program C++. Ini biasanya mencakup fitur seperti penyorotan sintaks, autocompletion, dan kemampuan navigasi cepat dalam kode untuk meningkatkan produktivitas.

- Manajemen Proyek: Dev C++ memungkinkan pengguna untuk membuat dan mengelola proyek-proyek C++. Ini melibatkan pengelompokan file sumber, mengonfigurasi pengaturan kompilasi, dan memudahkan manajemen file-file terkait proyek.

- Template Proyek: IDE ini menyediakan template proyek yang telah diatur sebelumnya untuk jenis-jenis aplikasi umum, seperti aplikasi konsol atau aplikasi GUI. Ini dapat membantu pemrogram memulai proyek dengan cepat tanpa perlu mengonfigurasi setiap detail proyek.

2. Compiler:

- Dev C++ menggunakan GCC (GNU Compiler Collection) sebagai compiler-nya. GCC adalah compiler yang sangat populer dan open source, mendukung banyak bahasa pemrograman termasuk C++.

- Compiler bertanggung jawab untuk menerjemahkan kode sumber C++ menjadi bahasa mesin atau kode biner yang dapat dieksekusi oleh komputer. Dev C++ menyertakan versi GCC yang diintegrasikan dengan IDE untuk menyederhanakan proses kompilasi.

3. Debugger:

- IDE ini juga menyertakan fasilitas debugging, yang memungkinkan pengguna untuk melacak eksekusi program, menemukan bug, dan memahami perilaku program pada saat runtime.

- Debugger memungkinkan langkah demi langkah eksekusi program, pemeriksaan nilai variabel, dan peninjauan stack trace, yang semuanya sangat berguna dalam mencari dan memperbaiki kesalahan dalam kode.

4. Siklus Pengembangan Perangkat Lunak:

- Proses pengembangan perangkat lunak umumnya melibatkan penulisan kode, kompilasi, debugging, dan pengujian. Dev C++ menyederhanakan langkah-langkah ini dengan menyatukannya dalam lingkungan yang terintegrasi.

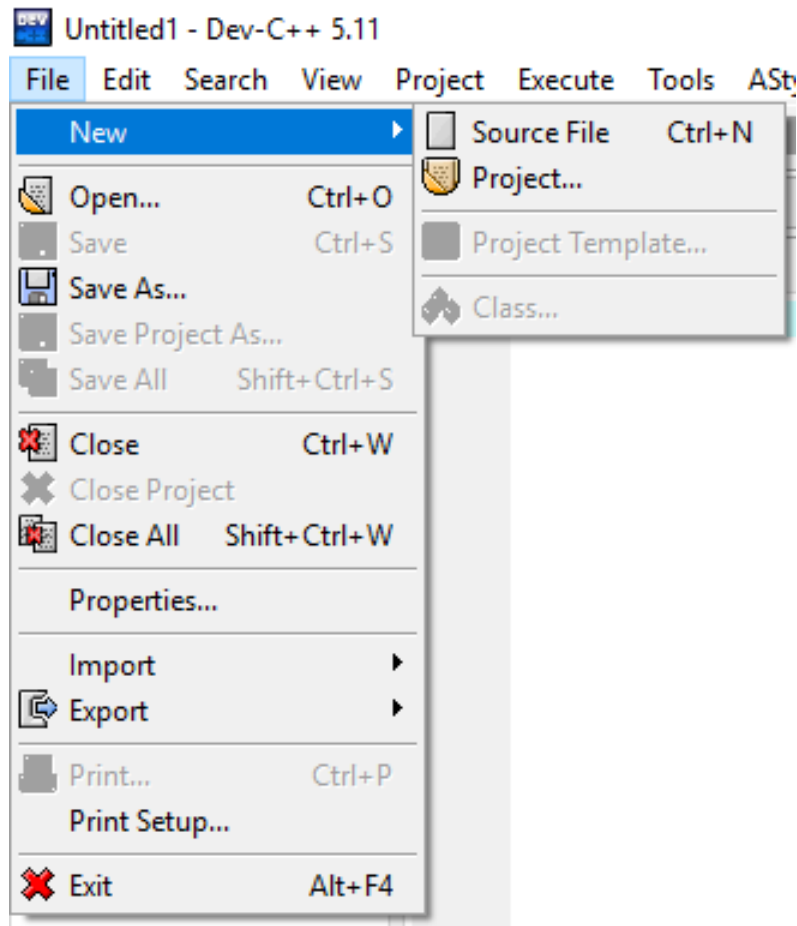
- Siklus pengembangan ini membantu pengembang untuk membuat, menguji, dan memelihara aplikasi C++ dengan lebih efisien.

5. Komunitas dan Dukungan:

- Dev C++ memiliki basis pengguna yang cukup besar dan forum-forum online yang dapat membantu pengguna untuk memecahkan masalah atau mendapatkan saran.

Menu pada Dev c++ yang sering digunakan beserta fungsinya

2.3 Menu File



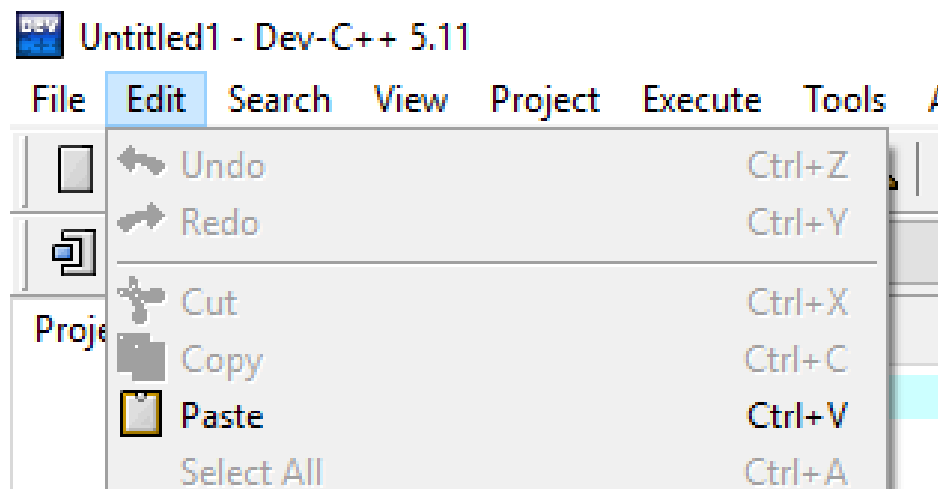
Gambar 2.2 Menu File

memiliki beberapa Submenu yaitu

- Source File: Untuk membuat atau membuka file baru atau yang telah ada dalam lingkungan pengkodean Dev C++.
- Project: Untuk membuat, membuka, atau mengelola proyek program yang sedang dikembangkan.
- Open: Memungkinkan pengguna untuk membuka file yang sudah ada dalam sistem.
- Save: Menyimpan perubahan pada file yang sedang aktif.
- Save As: Menyimpan file dengan nama baru atau lokasi baru.
- Save Project As: Menyimpan proyek dengan nama baru atau lokasi baru.

- Save All: Menyimpan semua perubahan pada file yang sedang aktif dalam proyek.
- Close: Menutup file yang sedang aktif.
- Close Project: Menutup proyek yang sedang aktif.
- Close All: Menutup semua file yang terbuka dalam lingkungan pengkodean.
- Properties: Memberikan akses ke properti atau pengaturan terkait file, proyek, atau lingkungan kerja.
- Import: Mengimpor file atau data dari lokasi lain ke dalam proyek yang sedang dibuat.
- Export: Mengirim file atau data dari proyek ke lokasi atau format lain.
- Print: Mencetak konten yang terdapat dalam file atau proyek.
- Print Setup: Mengatur pengaturan pencetakan sebelum mencetak.
- Exit: Menutup aplikasi Dev C++.

2.4 Menu Edit



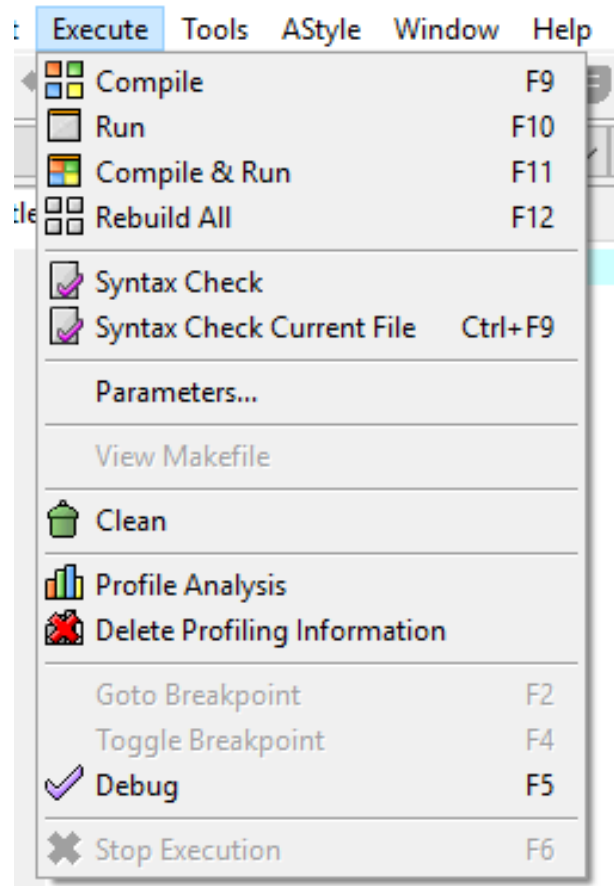
Gambar 2.3 Menu Edit

Menu Edit dalam Dev C++ menyediakan fungsi-fungsi penting untuk mengedit kode program. Berikut adalah penjelasan singkat mengenai fungsi-fungsi tersebut:

- Undo: Digunakan untuk membatalkan atau mengurangi perubahan terakhir yang dilakukan pada kode. Fungsi ini memungkinkan untuk kembali ke versi sebelumnya sebelum modifikasi terakhir.

- Redo: Melakukan kembali aksi yang telah dibatalkan dengan fungsi Undo. Jika pengguna telah menggunakan Undo, Redo memungkinkan untuk mengembalikan perubahan yang dibatalkan.
- Cut: Menghapus teks atau kode yang dipilih dan menempatkannya ke dalam clipboard. Setelah dipotong, konten dapat ditempatkan di lokasi lain dengan fungsi Paste
- . Copy: Menyalin teks atau kode yang dipilih ke dalam clipboard tanpa menghapusnya dari tempat asalnya. Konten yang disalin dapat ditempatkan di tempat lain dengan fungsi Paste.
- Paste: Memasukkan teks atau kode yang telah disalin atau dipotong dari clipboard ke lokasi yang dipilih dalam kode program.
- Select All: Memilih seluruh teks atau kode dalam jendela editor. Dengan menggunakan fungsi ini, semua isi dapat dipilih sekaligus untuk dieksekusi (misalnya, untuk copy, cut, atau edit).

2.5 Menu Execute



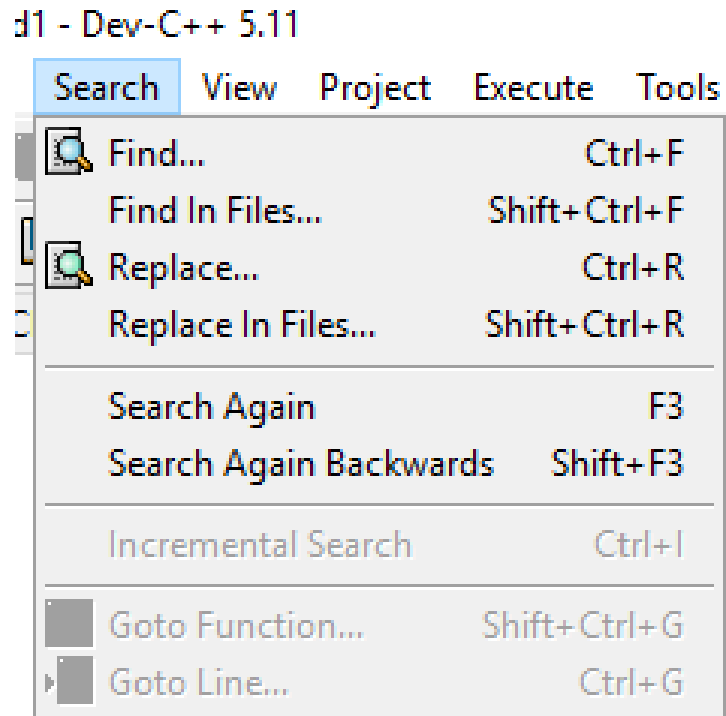
Gambar 2.4 Menu Tools

Fungsi Menu Tools pada Dev C++

- **Compile:** Menu Compile digunakan untuk mengonversi kode sumber yang telah ditulis ke dalam bahasa mesin agar dapat dieksekusi. Setelah dikompilasi, kode akan diperiksa untuk kesalahan sintaksis tetapi belum dieksekusi secara langsung.
- **Run:** Opsi Run di menu Tools digunakan untuk mengeksekusi program yang telah dikompilasi. Ini akan menjalankan program yang sedang aktif atau yang telah dikompilasi sebelumnya.
- **Compile and Run:** Fungsi Compile and Run melakukan kompilasi terhadap kode sumber dan secara otomatis menjalankan program yang dihasilkan tanpa perlu melakukan proses terpisah.
- **Rebuild All:** Opsi Rebuild All digunakan untuk membersihkan dan membangun ulang (rebuild) seluruh proyek. Ini akan memaksa Dev C++ untuk

membangun ulang seluruh proyek, bahkan jika kode sumber tidak berubah. Hal ini sering digunakan jika terdapat masalah pada kompilasi atau jika ingin memastikan semua perubahan diterapkan.

2.6 Menu Search



Gambar 2.5 Menu Search

Menu Search pada Dev C++ memiliki beberapa fungsi yang berguna dalam proses pengeditan kode yaitu :

- Find: Memungkinkan pencarian kata atau frasa tertentu dalam satu file kode. Pengguna dapat mencari kata kunci tertentu dalam dokumen yang sedang aktif.
- Find in Files: Melakukan pencarian untuk kata kunci atau frasa di seluruh proyek atau direktori yang dipilih. Ini membantu mencari di berbagai file sekaligus.
- Replace: Memungkinkan pengguna untuk menggantikan kata kunci tertentu dengan kata atau frasa lainnya dalam satu file kode.
- Replace in Files: Sama seperti Replace, tetapi berlaku untuk penggantian di seluruh proyek atau direktori yang dipilih.

- Search Again: Melanjutkan pencarian berikutnya dari kata kunci yang sebelumnya dicari.
- Search Again Backwards: Melakukan pencarian sebelumnya dari kata kunci yang telah dicari sebelumnya ke arah sebelumnya.
- Incremental Search: Memungkinkan pencarian real-time saat pengguna mengetikkan kata kunci. Ini secara otomatis menyoroti hasil pencarian saat pengguna mengetik.
- 8. Go to Function: Memungkinkan pengguna untuk melompat langsung ke fungsi tertentu dalam kode.
- Go to Line: Memungkinkan pengguna untuk langsung menuju baris kode tertentu dalam file yang sedang aktif.

BAB 3

JENIS-JENIS OPERATOR DALAM C++

3.1 Pengertian Operator Dalam C++

Dalam bahasa pemrograman terutama pasti tidak akan lepas dari yang namanya yaitu Operator, Operator dalam bahasa pemrograman sebenarnya hampir sama dengan operator dalam Matematika. Namun, Tentu saja menggunakan logo atau simbol yang berbeda

Operator dalam C++ dipakai sebagai simbol atau karakter yang dimasukkan kedalam program dengan tujuan melakukan sebuah manipulasi atau operasi pada nilai yang ingin kita manipulasi, Contoh sederhananya yaitu kita ingin Menaikkan 'Nilai' dari sebuah variabel dengan besar nilai itu 'ditambah' 1, Maka kita bisa menggunakan Operator + atau Tambah untuk memanipulasi nilai tersebut,

Setelah mengerti sedikit tentang pengertian dari sebuah operator, Mari langsung saja kita bahas apa saja jenis yang ada pada C++

Ada 5 Jenis Operator yang akan kita bahas yaitu:

- A. Operator Aritmatika**
- B. Operator Penugasan**
- C. Operator Pembandingan**
- D. Operator Logika**
- E. Operator Lainnya**

Mari kita bahas satu persatu Operator-Operator yang ada diatas:

3.2 Operator Aritmatika

Operator aritmatika ialah operator yang biasanya kita pakai pada saat belajar matematika, Contohnya seperti dasar matematika yaitu + (Tambah), - (Pengurangan), Dan lainnya, Mari kita bahas

Nama Operator	Simbol
Penjumlahan	+

UNIVERSITAS TEKNOKRAT INDONESIA

KAMPUSNYA SANG JUARA

Pengurangan	-
Perkalian	*
Pembagian	/
Sisa Bagi	%

Tabel 3.1 Operator aritmatika dalam C++

Bagaimana Cara Pakainya?

Mari kita langsung saja membuat kodenya kedalam program C++, Berikut adalah contoh untuk Operator (+)

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

int main(){

    int a, b, c;

    cout << "Inputkan nilai a: ";
    cin >> a;
    cout << "Inputkan nilai b: ";
    cin >> b;

    // menggunakan operator penjumlahan
    c = a + b;

    cout << "Hasil a + b = " << c << endl;

    return 0;
}
```

Gambar 3.1 Contoh Kode Aritmatika A.1

Maka Hasilnya akan menjadi

```
Inputkan nilai a: 5
Inputkan nilai b: 6
Hasil a + b = 11
-----
Process exited after 5.622 seconds with return value 0
Press any key to continue . . .
```

Gambar 3.2 Contoh hasil Kode Aritmatika A.2

Dan
semua operator aritmatika yang ada pada C++

Contoh diatas juga berlaku untuk

Berikut adalah contoh kodenya:

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

int main(){

    int a, b;

    cout << "Inputkan nilai a: ";
    cin >> a;

    cout << "Inputkan nilai b: ";
    cin >> b;

    cout << "Hasil a + b: " << a + b << endl;
    cout << "Hasil a - b: " << a - b << endl;
    cout << "Hasil a * b: " << a * b << endl;
    cout << "Hasil a / b: " << a / b << endl;
    cout << "Hasil a % b: " << a % b << endl;

    return 0;
}
```

Gambar 3.3 Contoh Semua Operator Aritmatika A.3

Maka Hasilnya:

```
Inputkan nilai a: 7
Inputkan nilai b: 2
Hasil a + b: 9
Hasil a - b: 5
Hasil a * b: 14
Hasil a / b: 3
Hasil a % b: 1

-----
Process exited after 3.735 seconds with return value 0
Press any key to continue . . .
```

Gambar 3.4 Contoh Hasil Kode A.4

Pada Operasi pembagian, $7/2$ hasilnya adalah 3, Mengapa demikian?, Hal tersebut terjadi karena kita menggunakan integer sebagai tipe data dari variabel a dan b, Untuk

mendapat hasil yang akurat dalam pembagian, kita bisa khusus kan untuk pembagian menggunakan 'Float'

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

int main(){

    float a, b;

    cout << "Inputkan nilai a: ";
    cin >> a;

    cout << "Inputkan nilai b: ";
    cin >> b;

    cout << "Hasil a / b: " << a / b << endl;

    return 0;
}
```

Gambar 3.5 Contoh Kode Aritmatika

Maka Hasilnya akan Seperti ini

```
Inputkan nilai a: 7
Inputkan nilai b: 2
Hasil a / b: 3.5

-----
Process exited after 2.666 seconds with return value 0
Press any key to continue . . .
```

Gambar 3.6 Contoh Hasil Kode Aritmatika A.6

Dan untuk Operator Modulus (%) Sendiri Yaitu operator Sisa bagi / modulus,yaitu sisa bagi dari pembagian,

Contoh:

4 / 2 = maka modulusnya/sisa baginya adalah 0 Tetapi Beda kalau,4 / 3 = maka modulusnya adalah 1

3.3 Operator Penugasan

Operator penugasan (Assignment Operator) merupakan operator untuk memberikan tugas pada variabel. Biasanya untuk mengisi nilai, Berikut adalah beberapa dari operator penugasan:

Nama Operator	=
---------------	---

UNIVERSITAS TEKNOKRAT INDONESIA

KAMPUSNYA SANG JUARA

Pengisian Dan Penambahan	+=
Pengisian Dan Pengurangan	-=
Pengisian Dan Pembagian	/=
Pengisian Dan Perkalian	*=
Pengisian Dan Sisa Bagi	%=

Tabel 3.2 Operator Penugasan

Berikut Adalah Contoh kode yang memakai Operator-operator diatas

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

int main(){

    int a, b;

    // pengisian nilai dengan operator =
    a = 5;
    b = 10;

    // pengisian sekaligus penambahan
    b += a; // ini sama seperti b = b + a
    cout << "Hasil b += a adalah " << b << endl;

    // pengisian sekaligus pengurangan
    b -= a; // ini sama seperti b = b - a
    cout << "Hasil b -= a adalah " << b << endl;

    // pengisian sekaligus perkalian
    b *= a; // ini sama seperti b = b * a
    cout << "Hasil b *= a adalah " << b << endl;

    // pengisian sekaligus pembagian
    b /= a; // ini sama seperti b = b / a
    cout << "Hasil b /= a adalah " << b << endl;

    // pengisian sekaligus penambahan
    b %= a; // ini sama seperti b = b % a
    cout << "Hasil b %= a adalah " << b << endl;

    return 0;
}
```

Maka Hasilnya: *Gambar 3.7 Contoh Kode penugasan*

```
Hasil b += a adalah 15
Hasil b -= a adalah 10
Hasil b *= a adalah 50
Hasil b /= a adalah 10
Hasil b %= a adalah 0

-----
Process exited after 0.01154 seconds with return value 0
Press any key to continue . . .
```

Gambar 3.8 Contoh Hasil Kode penugasan B.2

Pada Program Tersebut, Variabel b kita isi ulang dengan operator penugasan,

Sebagai Contoh

```
b += a
```

Yaitu bisa juga diartikan seperti

```
b = b + a
```

Hal ini juga berlaku kepada operator-operator yang lainnya seperti (-),(*),(/) Dan lainnya

3.4 Operator Pembandingan

Operator pembandingan adalah operator untuk memabandingkan dua buah nilai.

Operator ini juga dikenal dengan operator relasi.

Operator Pembandingan Terdiri dari:

Nama Operator	Simbol
Lebih Besar	>
Lebih Kecil	<
Sama Dengan	==
Tidak Sama Dengan	!=
Lebih Besar Sama Dengan	>=
Lebih Kecil Sama Dengan	<=

Tabel 3.3 Operator Pembandingan

Nilai yang dihasilkan dari operasi pembandingan akan berupa **true** dan **false**.

Pada Bahasa C++, Nilai **true** akan sama dengan 1 dan **false** akan sama dengan 0

Berikut adalah contoh kode nya

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

int main(){
    int a = 4, b = 3;
    bool hasil;

    cout << "a = " << a << endl;
    cout << "b = " << b << endl;

    // menggunakan operator pembandingan
    hasil = a > b;
    cout << "a > b = " << hasil << endl;

    hasil = a < b;
    cout << "a < b = " << hasil << endl;

    hasil = a >= b;
    cout << "a >= b = " << hasil << endl;

    hasil = a <= b;
    cout << "a <= b = " << hasil << endl;

    hasil = a == b;
    cout << "a == b = " << hasil << endl;

    hasil = a != b;
    cout << "a != b = " << hasil << endl;

    return 0;
}
```

M
aka

Gambar 3.9 Contoh Kode Pembandingan C.1

Hasilnya Adalah:

```
a = 4
b = 3
a > b = 1
a < b = 0
a >= b = 1
a <= b = 0
a == b = 0
a != b = 1

-----
Process exited after 0.01023 seconds with return value 0
Press any key to continue . . .
```

Gambar 3.10 Contoh Kode Pembandingan C.2

Penjelasannya yaitu $a > b$, a bernilai 4 dan b bernilai 3, Maka apakah benar(true) kalau $a(4)$ lebih besar dari $b(3)$, Jawabannya ialah true(1) Dan seterusnya

3.5 Operator Logika

Operator logika dipakai untuk menghasilkan nilai boolean true atau false dari 2 kondisi atau lebih.

Operator Logika juga terdiri dari :

Nama Operator	Simbol	Contoh
Logika AND	&&	true && false, hasilnya: false
Logika OR		true false, hasilnya: false
Not/Kebalikan	!	!false, hasilnya: true

Tabel 3.4 Operator Logika

Catatan: operator OR menggunakan karakter pipe " | ", bukan huruf L kecil. Karakter pipe ini bergabung dengan tombol "\" dan ditekan menggunakan tombol shift.

Seperti Pada Logika Berikut:

- Pernyataan 1: Apakah manusia perlu makanan
- Pernyataan 2: apakah manusia juga perlu untuk minum air

Maka kita akan dahulu mengecek kebenarannya

- Pernyataan 1: Apakah manusia perlu makanan = true
- Pernyataan 2: apakah manusia juga perlu untuk minum air = true

Maka pernyataan 1 && pernyataan 2 = true

Agar lebih paham mari kita bahas terlebih dahulu sifat dari ketiga operator yang akan kita bahas

- Operator && hanya akan menghasilkan true jika kedua operand bernilai true, selain itu hasilnya false.
- Operator || hanya akan menghasilkan false jika kedua operand bernilai false, selain itu hasilnya true .
- Operator ! Akan membalikkan logika, !false menjadi true, !true menjadi false.

Berikut adalah contoh kode program sederhana yang memakai operator logika

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      bool a = true;
8      bool b = false;
9      bool hasil;
10
11     hasil = a && a;
12     cout << "Hasil dari a && a : " << hasil << endl;
13
14     hasil = a && b;
15     cout << "Hasil dari a && b : " << hasil << endl;
16
17     hasil = a || b;
18     cout << "Hasil dari a || b : " << hasil << endl;
19
20     hasil = b || b;
21     cout << "Hasil dari b || b : " << hasil << endl;
22
23     hasil = !a;
24     cout << "Hasil dari !a : " << hasil << endl;
25
26     hasil = !b;
27     cout << "Hasil dari !b : " << hasil << endl;
28
29     return 0;
30 }
```

Gambar 3.11 Contoh Kode Logika D.1

Hasilnya Maka

```
Hasil dari a && a : 1
Hasil dari a && b : 0
Hasil dari a || b : 1
Hasil dari b || b : 0
Hasil dari !a : 0
Hasil dari !b : 1
```

Gambar 3.11 Contoh Kode Logika D.2

:

Penjelasan singkat dari kode diatas yaitu:

a=bernilai true(benar)

b=bernilai false(salah)

- ❖ Rumus &&(AND) ialah, Jika kedua kondisi sama sama benar, maka pernyataan/kondisi tersebut adalah 'benar', Jika Kondisi 'Benar' dan ada 'Salah'

Maka operator (AND) Akan Menyatakan ‘salah’ karena ada salah satu yang ‘salah’

- ❖ Rumus || (OR) ialah, Jika ‘Salah Satu’ Kondisi ‘benar’ Maka pernyataan tersebut benar
- ❖ Rumus ! (NOT) akan mem‘balikkan’ sifat dari suatu nilai, jika suatu kondisi, misal benar diberi ! maka akan menjadi !benar, Maka pernyataan tersebut akan di‘balikkan’ menjadi sebaliknya yaitu ‘salah’

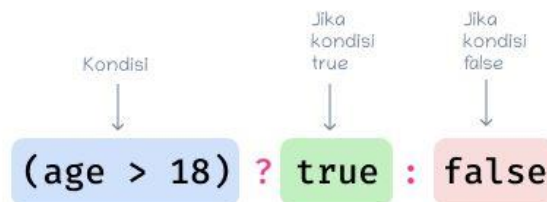
3.6 Operator Lainnya

Setelah kita membahas beberapa operator yang ada diatas, terdapat juga operator lain yang harus kita ketahui:

Nama Operator	Simbol	Keterangan
Ternary	? :	Untuk Membuat kondisi
Increment	++	Untuk menambah 1
decrement	--	Untuk mengurangi 1

Tabel 3.5 Operator Lain

Berikut ialah operator ternary untuk membuat kondisi:



Gambar 3.12 Contoh Kode Ternary E.1

Berikut Contoh kode nya:

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

int main(){
    int a = 4;

    // menggunakan operator ternary
    string hasil = a > 1 ? "benar": "salah";

    cout << "a > 1 adalah " << hasil << endl;

    return 0;
}
```

hasil outputnya

Gambar 3.13 Contoh Kode Ternary E.2

```
a > 1 adalah benar
```

Gambar 3.13 Contoh Kode Ternary E.3

Yang Terakhir yaitu ada operator increment dan decrement untuk menambah dan mengurangi nilai dengan **1**

Berikut adalah contoh kode nya:

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

int main(){
    int a = 4;

    // increment a
    a++;
    cout << "a++ = " << a << endl;

    // increment lagi a
    ++a;
    cout << "++a = " << a << endl;

    // decrement a
    a--;
    cout << "a-- = " << a << endl;

    // decrement lagi a
    --a;
    cout << "--a = " << a << endl;

    return 0;
}
```

Gambar 3.14 Contoh Kode Increment dan decrement

Dan Hasil Outputnya

```
a++ = 5
++a = 6
a-- = 5
--a = 4
```

Gambar 3.15 Contoh Hasil Kode increment dan decrement

Operator increment dan decrement sering kali ditaruh pada kode yang mengandung perulangan, Operator satu ini sangat berguna pada saat kita memakai kode yang bersifat Memiliki perulangan, Contoh pemakaiannya yaitu pada program parkir yang akan kita bahas nantinya di bab 4, dan bukan hal ini juga bukan hanya berlaku pada

program parkir, tapi program program yang Advanced atau ketingkat tinggi, increment maupun decrement sangat terpakai, tentunya hal ini tergantung oleh programer dan kebutuhan dari programnya seperti apa.

BAB 4

PERINTAH DASAR PEMROGRAMAN C++

4.1 Pendahuluan

Imajinasikanlah bahwa menulis program dalam bahasa C++ sama serunya seperti bercerita kepada teman robot yang menggemaskan. Di dunia coding ini, setiap perintah dasar seperti `#include <iostream>`, `using namespace std`, `int main()`, `cout`, `cin`, dan `return 0` seakan memberikan kekuatan ajaib dan membuka kamus rahasia agar bisa berkomunikasi dan berpetualang dengan teman robotmu dalam bahasa yang khusus mereka mengerti.

Ini benar-benar seperti memberikan buku petualangan magis kepada robot lucu tersebut. `#include <iostream>` adalah bagian pertama dari buku tersebut yang membuka gerbang petualangan, memungkinkan robot untuk berbicara dan mendengarkan dengan antusiasme yang menyala-nyala, Saat kita menggunakan `using namespace std`, hal itu seperti menemukan bab yang berisi kode rahasia dalam buku cerita tersebut, menjadikan robot itu cerdas seperti karakter-karakter dalam petualangan yang kita buat bersama-sama.

Sekarang, `int main()` menjadi panggung besar di mana semua petualangan dimulai. Di panggung ini, robot tersebut bertanya-tanya kepada kita dengan menggunakan `cout`, seolah-olah karakter dalam cerita yang aktif berinteraksi dengan kita dalam setiap adegan yang seru.

Sementara `cin` adalah cara cerdas robot itu mendengarkan setiap kata yang kita sampaikan dalam petualangan tersebut. Kita bisa memberikan arahan atau keputusan pada jalan cerita ini, membuatnya semakin menarik. Tambahkan `endl`, itu seperti memberikan jeda dramatis dalam cerita, menciptakan momen yang mendalam atau adegan yang tak terlupakan. Ini memberikan kesempatan bagi cerita untuk bernapas, memberikan kesan yang lebih kuat dan membuat petualangan menjadi lebih hidup.

Dan ketika kita sampai pada bagian `return 0`, itu seakan-akan memberi sinyal akhir pada babak petualangan itu. Kita menutup buku cerita sementara setelah petualangan yang luar biasa, namun dengan antusiasme menantikan babak selanjutnya dari kisah seru yang akan kita ciptakan bersama teman robot kita

UNIVERSITAS TEKNOKRAT INDONESIA

KAMPUSNYA SANG JUARA

berikut adalah contoh program yang mencakup penjelasan dalam setiap perintahnya sesuai dengan analogi sebelumnya:

```
[*] Untitled2
1  #include <iostream> // Buka pintu cerita
2
3  using namespace std; // Gunakan kamus rahasia
4
5  int main() { // Mulai cerita petualangan
6      cout << "Halo, penjelajah dunia baru!" << endl; // Sapa robot lucu kita dengan antusiasme
7
8      int usia; // Siapkan alat untuk memasukkan informasi
9      cout << "Berapa usia robot ini? Tulis dalam angka: "; // Tanya robot lucu kita
10     cin >> usia; // Dengarkan jawabannya
11
12     cout << "Wow! Robot ini berusia " << usia << " tahun dalam petualangan ini!" << endl; // Ceritakan kembali apa yang kita dengar
13
14     cout << "Sampai jumpa, teman robot! Petualangan kita akan terus berlanjut..." << endl; // Akhiri cerita petualangan dengan semangat
15
16     return 0; // Tutup buku cerita, sambil menanti babak selanjutnya
17 }
18
```

Gambar 4.1 Program Sesuai Cerita Diatas

Dalam penjelasan ini:

- Penjelasan:
- - `#include <iostream>` adalah seperti membuka pintu untuk memulai petualangan kita, memungkinkan kita menggunakan fungsi input-output untuk berinteraksi dengan robot kita.
- - `using namespace std;` mengizinkan kita untuk menggunakan kata kunci tanpa harus menambahkan `std::` sebelumnya, seperti membuat bahasa rahasia bersama robot kita.
- - `int main()` adalah panggung besar di mana cerita petualangan kita dimulai, di mana kita bisa berinteraksi dengan robot kita.
- - `cout << "Halo, penjelajah dunia baru!" << endl;` adalah cara kita menyapa robot kita dengan gembira, memberi semangat pada awal petualangan.
- - `int usia;` adalah alat yang kita siapkan untuk mendengarkan informasi. Ini adalah tempat di mana kita menyimpan usia yang diketahui dari robot.
- - `cout << "Berapa usia robot ini? Tulis dalam angka: ";` adalah pertanyaan kita kepada robot, seperti bertanya kepada karakter dalam cerita.
- - `cin >> usia;` adalah bagaimana kita mendengarkan jawaban dari robot kita dan menyimpannya.
- - `cout << "Wow! Robot ini berusia " << usia << " tahun dalam petualangan ini!" << endl;` adalah cara kita menceritakan kembali informasi yang telah kita terima dari robot, menambahkan kisah dalam petualangan.

- - `cout << "Sampai jumpa, teman robot! Petualangan kita akan terus berlanjut..."
<< endl;` adalah cara kita mengakhiri babak petualangan ini, tetapi dengan antusiasme yang tinggi untuk menanti petualangan berikutnya bersama robot kita.

Intinya ialah yaitu, Pada bab ini kita akan mempelajari fungsi fungsi atau perintah dasar yang ada pada C++ contohnya seperti analogi diatas yaitu `#include<iostream>` yaitu bisa dikatakan sebagai ‘perpustakaan’ karena memungkinkan program mengerti apa yang kita input/perintahkan, contoh yaitu agar program bisa menggunakan fungsi `cout` (mengeluarkan perintah yang kita masukkan) dan `cin` (menyimpan ‘nilai’ yang kita berikan) , mari kita pelajari lebih lanjut seperti yaitu `for`, `while` dan lainnya dihalaman berikut

4.2 Integer, Float Dan Double

Variabel dalam bahasa C++ bisa dibandingkan dengan kantong-kantong ajaib di dalam tas petualangan kita bersama teman robot yang penuh semangat. Setiap kantong ini memiliki bentuk dan ukuran yang berbeda, cocok untuk menyimpan barang-barang unik dalam cerita petualangan kita.

Variabel integer, seperti `int`, adalah kantong yang ideal untuk menyimpan angka bulat. Ini mirip dengan kantong yang kita gunakan untuk mengumpulkan berbagai macam barang di sepanjang perjalanan petualangan kita. Misalnya, jumlah permata yang ditemukan di dalam gua, atau jumlah teman yang kita jumpai dalam petualangan.

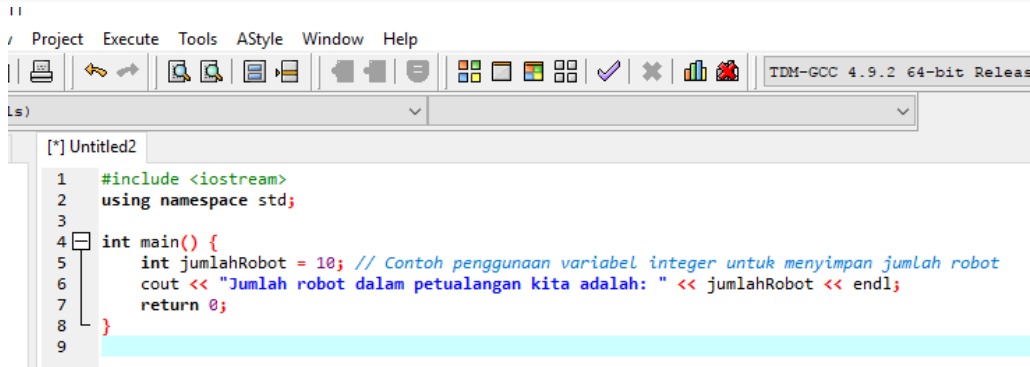
Sementara float dan double, seperti kantong dengan kemampuan istimewa, memungkinkan kita untuk menyimpan angka desimal. Mereka cocok untuk menyimpan informasi yang lebih spesifik, seperti jumlah uang yang berhasil kita kumpulkan selama petualangan atau mungkin tinggi yang tidak bulat persis dari menara megah yang kita temui.

Konstanta adalah petunjuk tak tergoyahkan dalam cerita kita. Mereka ibarat aturan dalam petualangan yang tak berubah. Misalnya, mungkin konstanta seperti petunjuk untuk menjaga harta karun tetap di tempatnya atau karakter utama yang memiliki sifat yang tidak berubah sepanjang cerita.

Dengan setiap kantong dan petunjuk ini, petualangan kita dalam bahasa C++ menjadi lebih kaya dan terorganisir. Mereka membantu kita menyimpan informasi yang penting dan memberikan arah yang jelas pada cerita yang kita tulis bersama teman robot kita. Jadi, dengan menggunakan variabel dan konstanta ini, kita bisa menyesuaikan petualangan kita dengan lebih baik dan membuat cerita yang semakin hidup dan menarik.

Berikut contohnya :

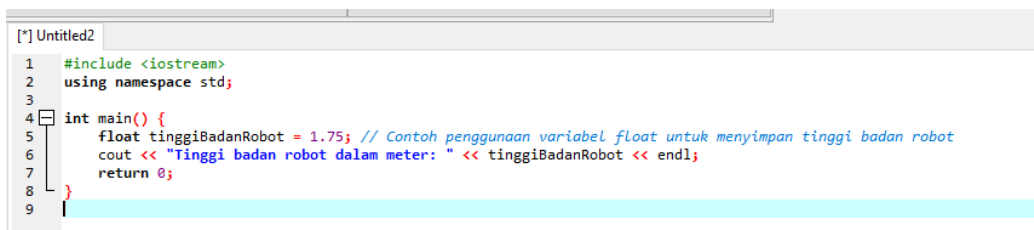
- A. Variabel Integer (`int`): Ini seperti tas kecil yang hanya bisa menampung angka bulat tanpa desimal.



Gambar 4.2 Program Yang menggunakan Integer

Di sini, jumlahRobot adalah variabel integer yang menyimpan nilai 10, yang merepresentasikan jumlah robot dalam petualangan kita.

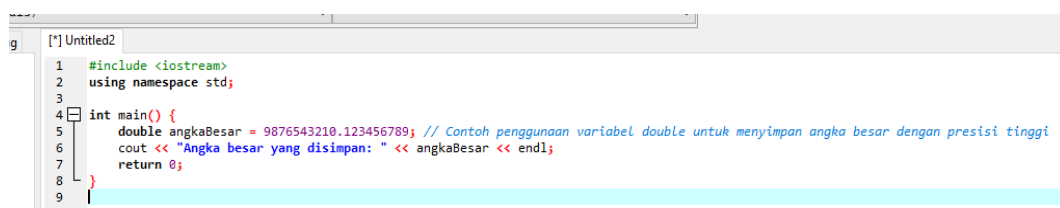
B. Variabel Float (float): Ini seperti tas yang bisa menampung angka desimal atau pecahan.



Gambar 4.3 Program Yang Menggunakan Float

Di sini, tinggiBadanRobot adalah variabel float yang menyimpan nilai 1.75, yang merupakan tinggi badan robot dalam meter dengan nilai desimal.

C. Variabel Double (double): Ini seperti tas super besar yang bisa menampung angka pecahan dengan presisi tinggi.



Gambar 4.4 Program Yang Menggunakan Double

Di sini, angkaBesar adalah variabel double yang menyimpan nilai yang sangat besar dengan presisi tinggi, yaitu 9876543210.123456789. Variabel ini cocok digunakan saat kita membutuhkan presisi yang sangat tinggi atau ketelitian dalam perhitungan.

D. variabel constanta (const): Konstanta adalah petunjuk tak tergoyahkan dalam cerita kita. Mereka ibarat aturan dalam petualangan yang tak berubah

```
[*] sadfadsadx.cpp
1  #include <iostream>
2  using namespace std;
3  int main() {
4      const int jumlahBuku = 10;
5      const float hargaBuku = 25.5;
6
7      cout << "Selamat datang di toko buku!" << endl;
8      cout << "Kami memiliki " << jumlahBuku << " buku tersedia." << endl;
9      cout << "Harga setiap buku adalah $" << hargaBuku << endl;
10
11     // Contoh penggunaan konstanta dalam operasi
12     float totalHarga = jumlahBuku * hargaBuku;
13     cout << "Total harga untuk semua buku adalah $" << totalHarga << endl;
14
15     return 0;
16 }
17
```

*Gambar 4.5 Program Yang Menggunakan
const/Konstanta*

4.3 String

Bayangkan jika kita ingin menyimpan nama-nama karakter dalam petualangan kita atau menyimpan pesan-pesan penting untuk diberikan kepada robot lucu kita. Nah, variabel string ini menjadi tas yang sangat fleksibel dan besar, mampu menyimpan teks apa pun yang kita inginkan.

Misalnya, jika kita ingin menyimpan nama karakter utama dalam petualangan kita, seperti "Bobby", "Alice", atau "Dora", kita akan meletakkannya dalam tas string ini. Selain itu, jika kita ingin menyimpan pesan yang panjang untuk robot kita, seperti "Halo, teman! Ayo kita mulai petualangan kita yang seru!", variabel string ini akan menjadi tempat yang sempurna untuk menyimpan pesan tersebut.

Variabel string ini sangat berguna karena tidak hanya mampu menyimpan kata-kata atau frasa, tetapi juga cerita atau teks yang lebih panjang. Sehingga, ketika kita ingin berkomunikasi atau menyampaikan informasi yang lebih kompleks kepada robot kita, kita dapat menggunakan tas ajaib ini untuk menyimpannya dengan rapi.

Dengan variabel string dalam C++, kita dapat menyimpan dan mengelola teks dengan mudah, memberikan dimensi yang lebih dalam pada petualangan kode kita dan memungkinkan kita untuk berinteraksi dengan teman robot kita dalam bahasa yang lebih kaya dan bermakna.

Berikut contohnya

```
[*] sadfdsadk.cpp
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      // Membuat tas ajaib string untuk menyimpan nama karakter utama
6      string namaKarakterUtama = "Bobby";
7
8      // Membuat tas ajaib string untuk menyimpan pesan awal petualangan
9      string pesanAwal = "Halo, teman! Ayo kita mulai petualangan kita yang seru!";
10
11     // Menggunakan tas-tas ajaib tersebut untuk berinteraksi dengan robot kita
12     cout << "Halo, robot! Nama karakter utama dalam petualangan kita adalah " << namaKarakterUtama << "." << endl;
13     cout << "Saatnya memulai perjalanan! " << pesanAwal << endl;
14
15     // Robot kita merespon dengan antusiasme
16     cout << "Robot: Wah, betapa menariknya! Ayo mulai petualangan kita!" << endl;
17
18     return 0;
19 }
20
```

Gambar 4.6 Program Yang Menggunakan String

Dalam contoh di atas, variabel `namaKarakterUtama` adalah tas ajaib string yang menyimpan nama karakter utama dalam petualangan kita, yaitu "Bobby". Variabel `pesanAwal` adalah tas ajaib string yang menyimpan pesan awal untuk memulai petualangan yang berisi "Halo, teman! Ayo kita mulai petualangan kita yang seru!".

Kemudian, kita menggunakan variabel-string tersebut (`namaKarakterUtama` dan `pesanAwal`) dalam `cout` untuk berkomunikasi dengan robot kita. Kita memberitahukan kepada robot nama karakter utama dan memulai petualangan dengan pesan yang kita simpan dalam tas ajaib string tadi.

Robot kita dengan antusiasme merespon permintaan kita, menandakan bagaimana variabel string ini memungkinkan kita untuk menyimpan dan berinteraksi dengan teks panjang dalam petualangan kode kita, membuatnya lebih hidup dan menarik bagi teman robot kita.

4.4 Boolean, If, If Else Dan If Else If

Variabel boolean dalam C++ seperti membawa bendera yang bisa memiliki dua kemungkinan, seperti bendera yang bisa dikibarkan untuk menunjukkan kebenaran atau kebohongan. Variabel ini bisa jadi seperti alat deteksi dalam cerita petualangan kita yang dapat memberi tahu robot kita apakah suatu kondisi itu benar atau salah.

Bayangkan jika kita ingin memberi tahu robot kita apakah cuaca cerah atau hujan dalam petualangan kita. Variabel boolean akan menjadi bendera yang kita kibarkan, mungkin bernama `cuacaCerah = true` jika memang cuacanya cerah atau `cuacaCerah = false` jika cuacanya hujan.

Sekarang, mari masuk ke dalam kondisi cerita yang lebih kompleks dengan menggunakan `if`, `if else`, dan `if else if`. Ini seperti bercerita kepada robot kita bagaimana ia harus bertindak tergantung pada situasi atau kondisi yang terjadi dalam petualangan kita.

Misalnya, jika kita ingin memberi tahu robot kita untuk membawa payung jika cuacanya hujan, kita akan menggunakan `if`. Jadi, "Jika cuaca hujan, bawa payung."

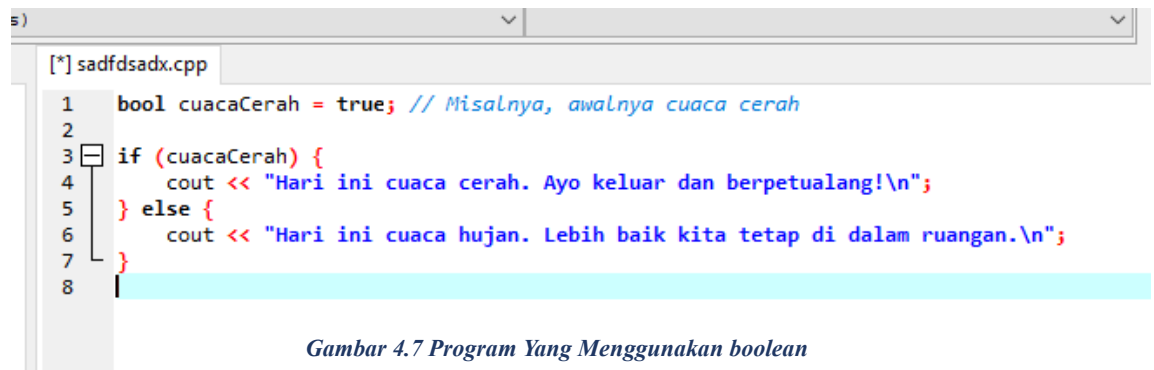
Kemudian, jika kita ingin memberitahu robot kita untuk tetap di dalam ruangan jika cuacanya hujan, tetapi keluar jika cerah, kita akan menggunakan `if else`. Artinya, "Jika cuaca hujan, tetap di dalam. Jika tidak, keluar dan berpetualang!"

Sementara itu, jika kita ingin memberitahu robot kita bahwa jika cuacanya hujan, bawa payung, tetapi jika cerah, bawa topi, itu seperti menggunakan `if else if`. Misalnya, "Jika cuaca hujan, bawa payung. Jika cerah, bawa topi untuk melindungi dari matahari."

Jadi, variabel boolean serta `if`, `if else`, dan `if else if` dalam C++ adalah cara bagi kita untuk memberi tahu robot kita bagaimana ia harus bertindak berdasarkan pada situasi atau kondisi yang terjadi dalam perjalanan petualangan kode kita. Analogi ini memungkinkan kita untuk mengatur jalannya cerita dengan lebih fleksibel dan dinamis, memberikan petualangan yang lebih seru dan menarik bagi robot kita.

Berikut contohnya :

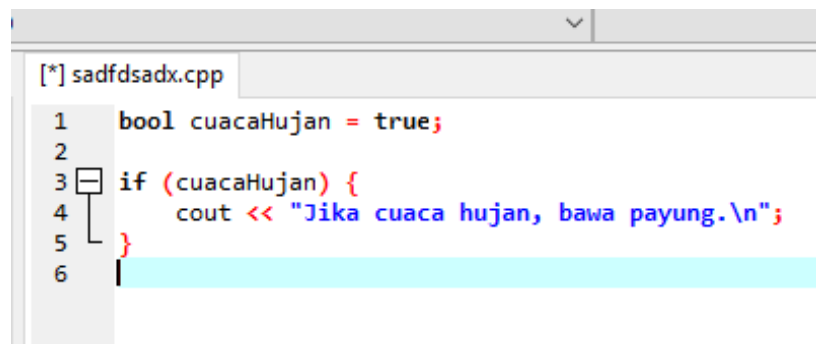
A. Boolean



Gambar 4.7 Program Yang Menggunakan boolean

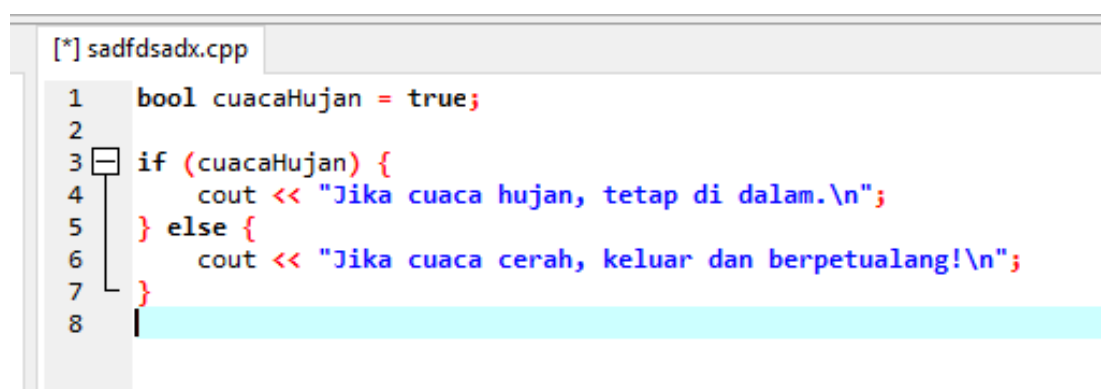
Dalam analogi tentang cuaca (cerah atau hujan), kita bisa menggunakan variabel boolean untuk menunjukkan keadaan cuaca

B. If



Gambar 4.8 Program Yang Menggunakan If

C. If Else



Gambar 4.9 Program Yang Menggunakan If Else

D. if else if

```
[*] sadfdsadx.cpp
1  bool cuacaHujan = true;
2  bool cuacaCerah = false;
3
4  if (cuacaHujan) {
5      cout << "Jika cuaca hujan, bawa payung.\n";
6  } else if (cuacaCerah) {
7      cout << "Jika cuaca cerah, bawa topi untuk melindungi dari matahari.\n";
8  }
9
```

Gambar 4.10 Program Yang Menggunakan If else If

4.5 For, While, Do While Dan Case Of

Bayangkan jika dalam petualangan kita, kita ingin mengulangi tugas tertentu beberapa kali atau melakukan sesuatu berulang kali dalam kondisi yang berbeda. Nah, di sinilah peran `for`, `while`, dan `do while` masuk ke dalam cerita.

Pertama, mari bicara tentang `'for'`. Ini seperti meminta robot kita untuk melakukan suatu tugas selama jumlah langkah tertentu dalam petualangan kita. Misalnya, jika kita ingin robot kita berjalan sejauh 10 langkah, kita bisa memberi instruksi: "Berjalanlah sebanyak 10 langkah".

Kemudian, `'while'` dan `'do while'` mirip dengan situasi di mana kita memberi tahu robot kita untuk melakukan tugas tertentu selama kondisi tertentu masih terpenuhi. Ini seperti memberi perintah pada robot kita untuk berjalan terus selama cuaca masih cerah. Jika cuaca cerah, robot kita akan terus berjalan. Jika `'while'`, itu seperti, "Selama cuaca cerah, terus berjalanlah." Sedangkan `'do while'`, meskipun cuacanya mungkin tidak cerah dari awal, tapi robot tetap akan berjalan setidaknya sekali sebelum memeriksa kondisi cuaca lagi. Misalnya, "Lakukan perjalanan pertama, lalu terus berjalan selama cuaca masih cerah."

Terakhir, ada `'switch-case'`. Ini adalah cara untuk memberi tahu robot kita untuk bertindak berdasarkan pada nilai atau situasi tertentu. Ini mirip dengan memberikan pilihan-pilihan pada robot kita dan meminta mereka untuk bertindak sesuai pilihan tersebut. Misalnya, jika kita memberikan pilihan pada robot kita antara berjalan, berlari, atau melompat, `'switch-case'` akan membantu kita untuk memberi instruksi berdasarkan pilihan yang robot kita ambil. Misalnya, "Jika memilih berjalan, lakukan perjalanan. Jika memilih berlari, lakukan lari cepat. Jika memilih melompat, lakukan lompatan tinggi."

Jadi, `for`, `while`, `do while`, dan `switch-case` dalam C++ membantu kita mengontrol bagaimana robot kita akan berinteraksi, bertindak, atau melakukan tugas-tugas tertentu dalam petualangan kode kita. Analogi ini memungkinkan kita untuk mengatur jalannya cerita dengan lebih terstruktur, memberikan petualangan yang lebih bervariasi dan dinamis bagi robot kita.

Berikut contohnya :

A. for

```
[*] sadfdsadx.cpp
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6      // Meminta robot untuk berjalan sebanyak 5 Langkah
7      for (int langkah = 1; langkah <= 5; ++langkah) {
8          cout << "Langkah ke-" << langkah << endl;
9      }
10     return 0;
11 }
12
```

Gambar 4.11 Program Yang Menggunakan For

Dalam contoh ini, loop for menginstruksikan robot untuk melakukan tugasnya (mencetak langkah-langkah) sebanyak 5 kali.

B. While

```
[*] sadfdsadx.cpp
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6      bool cuacaCerah = true;
7
8      // Robot akan terus berjalan selama cuaca cerah
9      while (cuacaCerah) {
10         cout << "Berjalan karena cuaca cerah!" << endl;
11         // Proses pembaruan cuaca (di dunia nyata akan bergantung pada kondisi aktual)
12         cuacaCerah = false; // Misalnya, mengubah cuaca menjadi tidak cerah
13     }
14     cout << "Cuaca tidak lagi cerah. Berhenti berjalan." << endl;
15     return 0;
16 }
17
```

Gambar 4.12 Program Yang Menggunakan While

Dalam contoh ini, loop while menunjukkan robot berjalan selama kondisi cuacaCerah masih benar (atau true). Setelah kondisi berubah menjadi salah (atau false), robot berhenti berjalan.

C. Do while

```
[*] sadfdsadx.cpp
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6      bool cuacaCerah = false;
7
8      // Robot akan melakukan perjalanan setidaknya sekali, kemudian terus berjalan selama cuaca cerah
9      do {
10         cout << "Melakukan perjalanan pertama." << endl;
11         // Proses pembaruan cuaca
12         cuacaCerah = true; // Misalnya, mengubah cuaca menjadi cerah
13     } while (cuacaCerah);
14
15     cout << "Sekarang cuaca cerah. Lanjutkan berjalan." << endl;
16     return 0;
17 }
18
```

Gambar 4.13 Program Yang Menggunakan Do While

Dalam contoh ini, loop do while menunjukkan robot melakukan perjalanan pertama tanpa memeriksa kondisi, dan kemudian berlanjut berjalan selama kondisi cuacaCerah benar (atau true).

D. Case of

```
LSJ
[*] sadfdsadx.cpp
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6      char pilihan = 'l'; // Misalnya, pilihan yang diambil adalah 'l'
7
8      switch (pilihan) {
9          case 'j':
10             cout << "Robot memilih untuk berjalan." << endl;
11             break;
12          case 'l':
13             cout << "Robot memilih untuk berlari." << endl;
14             break;
15          case 'm':
16             cout << "Robot memilih untuk melompat." << endl;
17             break;
18          default:
19             cout << "Pilihan tidak valid." << endl;
20      }
21
22     return 0;
23 }
24
```

Gambar 4.14 Program Yang Menggunakan Case Of

Dalam contoh ini, struktur switch-case memberikan pilihan pada robot untuk melakukan tindakan berdasarkan pilihan yang dipilih. Dalam kasus ini, robot memilih untuk berlari berdasarkan pilihan yang diberikan. Jika pilihan tidak ada pada kasus yang ada, program akan menjalankan blok default.

4.6 Array

Bayangkan jika kita memiliki rak buku di dalam perpustakaan yang memiliki banyak laci. Sekarang, setiap laci ini adalah tempat untuk menyimpan buku-buku dengan judul yang berbeda. Nah, dalam dunia C++, array mirip seperti rak buku ini, di mana kita dapat menyimpan sejumlah nilai atau data dalam satu tempat yang terstruktur.

Jadi, array adalah kumpulan variabel yang memiliki tipe yang sama dan dikelompokkan bersama di dalam satu tempat. Analogi rak buku ini membantu kita memahami bahwa array itu seperti kumpulan laci, dan setiap laci mewakili satu nilai atau elemen dalam array.

Misalnya, jika kita ingin menyimpan nilai-nilai suhu harian selama seminggu, kita bisa menggunakan array. Ini seperti memiliki tujuh laci di rak buku kita, masing-masing laci berisi nilai suhu harian untuk setiap hari dalam seminggu.

Array memungkinkan kita untuk mengakses nilai-nilai ini dengan cara yang terstruktur menggunakan indeks. Indeks dalam array adalah seperti nomor laci dalam rak buku; membantu kita menemukan nilai atau data yang spesifik dalam koleksi tersebut. Misalnya, jika kita ingin mengetahui suhu pada hari ke-3, kita bisa membuka laci yang sesuai dengan indeks ke-3 pada rak buku (atau dalam hal ini, indeks ke-2 karena indeks dimulai dari 0), dan kita akan menemukan nilai suhu untuk hari itu.

Jadi, dengan analogi rak buku ini, array dalam C++ memungkinkan kita untuk menyimpan sejumlah besar nilai atau data dalam satu tempat yang terorganisir dengan baik, mempermudah akses dan pengelolaan informasi.

Berikut contohnya :

```
[*] sadfadsadx.cpp
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      // Mendeklarasikan array untuk menyimpan suhu harian selama seminggu (7 hari)
6      int suhuHarian[7]; // Array dengan 7 elemen, mewakili 7 hari dalam seminggu
7
8      // Memasukkan nilai suhu harian ke dalam array
9      suhuHarian[0] = 28; // Hari ke-1
10     suhuHarian[1] = 30; // Hari ke-2
11     suhuHarian[2] = 27; // Hari ke-3
12     suhuHarian[3] = 29; // Hari ke-4
13     suhuHarian[4] = 25; // Hari ke-5
14     suhuHarian[5] = 26; // Hari ke-6
15     suhuHarian[6] = 31; // Hari ke-7
16
17     // Menampilkan nilai suhu harian untuk setiap hari dalam seminggu
18     cout << "Suhu harian selama seminggu:\n";
19     for (int i = 0; i < 7; ++i) {
20         cout << "Hari ke-" << i + 1 << ": " << suhuHarian[i] << " derajat Celsius" << endl;
21     }
22
23     return 0;
24 }
25
```

Gambar 4.15 Program Yang Menggunakan Array

Program di atas mendeklarasikan sebuah array `suhuHarian` yang memiliki 7 elemen, mewakili suhu harian selama seminggu. Kemudian, nilai suhu harian dimasukkan ke dalam array tersebut. Selanjutnya, program menampilkan nilai suhu harian untuk setiap hari dalam seminggu dengan menggunakan perulangan `for` untuk mengakses setiap elemen array.

Dalam analogi dengan rak buku, array `suhuHarian` adalah seperti rak buku dengan tujuh laci, masing-masing laci berisi nilai suhu untuk setiap hari dalam seminggu. Indeks dari 0 hingga 6 mewakili nomor laci dalam rak buku yang sesuai dengan hari ke-1 hingga hari ke-7.

Semoga analogi ini membantu dalam memahami bagaimana array digunakan dalam C++ untuk menyimpan dan mengakses sejumlah besar nilai atau data dalam satu tempat yang terstruktur.

BAB 5

MENGANALISIS PROSES PADA PROGRAM PARKIR

Terkhusus untuk bab ini kita akan mendedikasinya untuk menganalisis bagaimana cara kerja/proses dari program parkir yang akan dibagi menjadi 7 bagian sebagai berikut:

[1]

```
1  #include <iostream>
2  #include <string>
3  #include <ctime>
4
5  using namespace std;
6
7  const int MAKS_SLOT_MOTOR = 30;
8  const int MAKS_SLOT_MOBIL = 15;
9  const int TARIF_JAM_PERTAMA_MOTOR = 2000;
10 const int TARIF_JAM_PERTAMA_MOBIL = 5000;
```

Gambar 5.1 Bagian 1

[2]

```
12 int main() {
13     string nomorPlatMotor[MAKS_SLOT_MOTOR];
14     time_t waktuMasukMotor[MAKS_SLOT_MOTOR];
15     int waktuParkirMotor[MAKS_SLOT_MOTOR];
16     int jumlahMotor = 0;
17
18     string nomorPlatMobil[MAKS_SLOT_MOBIL];
19     time_t waktuMasukMobil[MAKS_SLOT_MOBIL];
20     int waktuParkirMobil[MAKS_SLOT_MOBIL];
21     int jumlahMobil = 0;
```

Gambar 5.2 Bagian 2

[3]

```
23 while (true) {
24     int pilihan;
25     cout << "Selamat\n";
26     cout << "datang"<<endl;
27
28     cout << "+-----+\n";
29     cout << "|      Selamat Datang di      |\n";
30     cout << "|      Tempat Parkir         |\n";
31     cout << "+-----+\n";
32     cout << "| Pilih jenis kendaraan:    |\n";
33     cout << "| 1. Motor                  |\n";
34     cout << "| 2. Mobil                  |\n";
35     cout << "| 3. Keluar                 |\n";
36     cout << "| 4. Info Parkir            |\n";
37     cout << "+-----+\n";
38     cout << "Pilihan: ";
39     cin >> pilihan;
```

Gambar 5.3 Bagian 3

[4]

```

41 switch (pilihan) {
42     case 1:
43         if (jumlahMotor < MAKS_SLOT_MOTOR) {
44             string plat;
45             cout << "Masukkan nomor identitas motor: ";
46             cin >> plat;
47
48             time_t waktuMasuk = time(0);
49             nomorPlatMotor[jumlahMotor] = plat;
50             waktuMasukMotor[jumlahMotor] = waktuMasuk;
51
52             cout << "Motor berhasil masuk. Nomor tiket: " << jumlahMotor + 1 << endl;
53             jumlahMotor++;
54         } else {
55             cout << "Tempat parkir motor penuh!" << endl;
56         }
57         break;
58     case 2:
59         if (jumlahMobil < MAKS_SLOT_MOBIL) {
60             string plat;
61             cout << "Masukkan nomor identitas mobil: ";
62             cin >> plat;
63
64             time_t waktuMasuk = time(0);
65             nomorPlatMobil[jumlahMobil] = plat;
66             waktuMasukMobil[jumlahMobil] = waktuMasuk;
67
68             cout << "Mobil berhasil masuk. Nomor tiket: " << jumlahMobil + 1 << endl;
69             jumlahMobil++;
70         } else {
71             cout << "Tempat parkir mobil penuh!" << endl;
72         }
73         break;

```

[5]

Gambar 5.4 Bagian 4

```

80 if (pilihan == 1 && jumlahMotor > 0) {
81     cout << "Masukkan nomor tiket motor yang keluar: ";
82     cin >> tiketKeluar;
83
84     if (tiketKeluar > 0 && tiketKeluar <= jumlahMotor) {
85         string plat = nomorPlatMotor[tiketKeluar - 1];
86         time_t waktuKeluar = time(0);
87         int waktuTotal = static_cast<int>(difftime(waktuKeluar, waktuMasukMotor[tiketKeluar - 1]) / 60.0);
88
89         cout << "Waktu masuk: " << ctime(&waktuMasukMotor[tiketKeluar - 1]);
90         cout << "Waktu keluar: " << ctime(&waktuKeluar);
91         cout << "Waktu parkir motor: " << waktuTotal << " menit" << endl;
92
93         int biayaTotal = TARIF_JAM_PERTAMA_MOTOR;
94         waktuTotal -= 60;
95         int tarifJamBerikutnya = TARIF_JAM_PERTAMA_MOTOR;
96         while (waktuTotal > 0) {
97             biayaTotal += tarifJamBerikutnya;
98             waktuTotal -= 60;
99         }
100         cout << "Biaya Parkir: Rp " << biayaTotal << endl;
101         cout << "-----\n";
102
103         for (int i = tiketKeluar - 1; i < jumlahMotor - 1; ++i) {
104             nomorPlatMotor[i] = nomorPlatMotor[i + 1];
105             waktuMasukMotor[i] = waktuMasukMotor[i + 1];
106             waktuParkirMotor[i] = waktuParkirMotor[i + 1];
107         }
108         jumlahMotor--;
109
110     } else {
111         cout << "Nomor tiket tidak ditemukan!" << endl;
112     }

```

Gambar 5.5 Bagian 5.1

```
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
...

case 4:
    cout << "\nInfo Parkir Motor:\n";
    for (int i = 0; i < jumlahMotor; ++i) {
        cout << "Nomor Tiket: " << i + 1 << endl;
        cout << "Nomor Identitas: " << nomorPlatMotor[i] << endl;
        cout << "Waktu Masuk: " << ctime(&waktuMasukMotor[i]);
    }

    time_t waktuSekarang = time(0); // Waktu saat ini
    int waktuParkir = static_cast<int>(difftime(waktuSekarang, waktuMasukMotor[i]) / 60.0); // Convert to minutes

    cout << "Waktu Parkir: " << waktuParkir << " menit" << endl;
    cout << "-----\n";
}

cout << "\nInfo Parkir Mobil:\n";
for (int i = 0; i < jumlahMobil; ++i) {
    cout << "Nomor Tiket: " << i + 1 << endl;
    cout << "Nomor Identitas: " << nomorPlatMobil[i] << endl;
    cout << "Waktu Masuk: " << ctime(&waktuMasukMobil[i]);
}

time_t waktuSekarang = time(0); // Waktu saat ini
int waktuParkir = static_cast<int>(difftime(waktuSekarang, waktuMasukMobil[i]) / 60.0); // Convert to minutes

cout << "Waktu Parkir: " << waktuParkir << " menit" << endl;
cout << "-----\n";
}
break;
Default:
cout << "Pilihan tidak valid." << endl;
break;
}
```

Gambar 5.6 Bagian 5.2

[6]

```
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
...

} else if (pilihan == 2 && jumlahMobil > 0) {
    cout << "Masukkan nomor tiket mobil yang keluar: ";
    cin >> tiketKeluar;

    if (tiketKeluar > 0 && tiketKeluar <= jumlahMobil) {
        string plat = nomorPlatMobil[tiketKeluar - 1];
        time_t waktuKeluar = time(0);
        int waktuTotal = static_cast<int>(difftime(waktuKeluar, waktuMasukMobil[tiketKeluar - 1]) / 60.0);

        cout << "Waktu masuk: " << ctime(&waktuMasukMobil[tiketKeluar - 1]);
        cout << "Waktu keluar: " << ctime(&waktuKeluar);
        cout << "Waktu parkir mobil: " << waktuTotal << " menit" << endl;

        int biayaTotal = TARIF_JAM_PERTAMA_MOBIL;
        waktuTotal -= 60;
        int tarifJamBerikutnya = TARIF_JAM_PERTAMA_MOBIL;
        while (waktuTotal > 0) {
            biayaTotal += tarifJamBerikutnya;
            waktuTotal -= 60;
        }

        cout << "Biaya Parkir: Rp " << biayaTotal << endl;
        cout << "-----\n";

        for (int i = tiketKeluar - 1; i < jumlahMobil - 1; ++i) {
            nomorPlatMobil[i] = nomorPlatMobil[i + 1];
            waktuMasukMobil[i] = waktuMasukMobil[i + 1];
            waktuParkirMobil[i] = waktuParkirMobil[i + 1];
        }
        jumlahMobil--;
    } else {
        cout << "Nomor tiket tidak ditemukan!" << endl;
    }
} else {
    cout << "Tidak ada kendaraan yang tersedia untuk dikeluarkan!" << endl;
}
break;
```

Gambar 5.7 Bagian 6

[7]

```
183 cout << "\n+-----+\n";
184 cout << "| Jumlah kendaraan yang masuk: |\n";
185 cout << "| Motor: " << jumlahMotor << " dari " << MAKS_SLOT_MOTOR << " |\n";
186 cout << "| Mobil: " << jumlahMobil << " dari " << MAKS_SLOT_MOBIL << " |\n";
187 cout << "+-----+\n";
188 }
189
190 return 0;
191 }
```

Gambar 5.8 Bagian 7

Mari Kita Analisis 7 Bagian tersebut

5.1 Bagian [1]

```
#include <iostream>
#include <string>
#include <ctime>
```

Yang dimaksud dari KodeTersebut adalah, Yaitu ada 3 Perpustakaan <iostream>, <string>, dan <ctime>

- <iostream> memungkinkan program untuk berkomunikasi dengan pengguna atau sistem melalui operasi input dan output standar. Contohnya adalah “cout: dan “cin”
- <string> Tipe data string menyediakan banyak fungsi dan metode yang mempermudah manipulasi teks, seperti menggabungkan string, mencari panjang string, dan lainnya. Dalam kode diatas “string digunakan untuk menyimpan nomor plat kendaraan
- <ctime> adalah bagian dari C Standard Library yang menyediakan fungsi-fungsi terkait waktu dan tanggal. Dalam kode Datas berikan, ctime digunakan untuk mengakses fungsi-fungsi seperti time() yang memberikan waktu saat ini sebagai nilai bertipe time_t, dan ctime(&waktu) yang mengonversi nilai waktu ke bentuk string yang dapat dibaca manusia.

`using namespace std;` = Pernyataan using namespace std, digunakan untuk mempermudah penggunaan elemen-elemen dari ruang nama (namespace) std tanpa harus menyertakan awalan std:: setiap kali menggunakannya, Dengan menggunakan using namespace std, kita dapat menghindari penulisan repetitif dari awalan std::, membuat kode lebih ringkas dan lebih mudah dibaca.

```
const int MAKS_SLOT_MOTOR = 30;
const int MAKS_SLOT_MOBIL = 15;
const int TARIF_JAM_PERTAMA_MOTOR = 2000;
const int TARIF_JAM_PERTAMA_MOBIL = 5000;
```

Mendefinisikan konstanta dan variabel yang akan digunakan dalam program, seperti batas maksimum slot parkir, tarif parkir per jam, dan tarif per menit tambahan. Dan **const** yaitu berarti variabel tersebut memiliki nilai tetap dan sudah bertipe data yaitu bilangan bulat (**integer**)

5.2 Bagian [2]

```
Int main(){  
    string nomorPlatMotor[MAKS_SLOT_MOTOR];  
    time_t waktuMasukMotor[MAKS_SLOT_MOTOR];  
    int waktuParkirMotor[MAKS_SLOT_MOTOR];  
    int jumlahMotor = 0;  
    string nomorPlatMobil[MAKS_SLOT_MOBIL];  
    time_t waktuMasukMobil[MAKS_SLOT_MOBIL];  
    int waktuParkirMobil[MAKS_SLOT_MOBIL];  
    int jumlahMobil = 0;
```

Membuat array dan variabel untuk menyimpan informasi kendaraan, seperti nomor plat, waktu masuk, waktu parkir, dan jumlah kendaraan.

`string nomorPlatMotor[MAKS_SLOT_MOTOR];` = adalah tipe data yaitu array, membuat variabel (`MAKS_SLOT_MOTOR`) Bisa diisi sampai berisikan 30 isi/nilai (dilihat dari inisialisasi variabel `const` pada [Bagian 1]), Variabel ini bisa disebut sebagai “Tempat Parkir” Dengan Kapasitas untuk motor 30 dan mobil 15

`time_t waktuMasukMotor` = Yaitu tipe data yang berasal dari Library/Perputakaan `<ctime>`, variabel ini berfungsi untuk menyamakan waktu antara program dan komputer, Dan tujuan dibuat seperti diatas karena agar program bisa mendapatkan waktu masuk “*Real Time*” dari kendaraan yang masuk

`int waktuParkirMotor` = menyimpan nilai waktu menjadi integer agar nantinya bisa dijadikan menjadi Menit

`int jumlahMotor = 0` = Variabel yang berfungsi untuk menjadi ‘wadah’ menyimpan informasi berapa banyak variabel kendaraan yang sudah masuk ke variabel `[MAKS_SLOT_MOTOR]` atau bisa dikatakan yaitu ‘tempat parkir’

5.3 Bagian [3]

`while (true) {` Membuat loop utama yang berjalan selama kondisi true. Program akan terus berjalan dan menunggu input pengguna.

```
int pilihan;
cout << "+-----+\n";
cout << "|  Selamat Datang di  |\n";
cout << "|    Tempat Parkir   |\n";
// ...
cout << "+-----+\n";
cout << "Pilihan: ";
cin >> pilihan;
```

Menampilkan menu utama kepada pengguna dan meminta input pilihan.

```
+-----+
a  Selamat Datang di  a
a    Tempat Parkir   a
a-----+
a Pilih jenis kendaraan: a
a 1. Motor           a
a 2. Mobil           a
a 3. Keluar           a
a 4. Info Parkir      a
+-----+
```

Gambar 5.9 Output 1

5.4 Bagian [4]

`switch (pilihan) {` = Menggunakan struktur switch-case untuk memproses dan memberikan pilihan menu kepada pengguna.

```
case 1:
    if (jumlahMotor < MAKS_SLOT_MOTOR) {
        string nomorPlat;
        cout << "Masukkan nomor identitas motor: ";
        cin >> nomorPlat;
        time_t waktuMasuk = time(0);
        nomorPlatMotor[jumlahMotor] = nomorPlat;
        waktuMasukMotor[jumlahMotor] = waktuMasuk;

        cout << "Motor berhasil masuk. Nomor tiket: " << jumlahMotor + 1 << endl;
        jumlahMotor++;
    } else {
        cout << "Tempat parkir motor penuh!" << endl;
    }
    break;
```

```
case 1:
    if (jumlahMotor < MAKS_SLOT_MOTOR) {
        string nomorPlat;
        cout << "Masukkan nomor identitas motor: ";
```

```
cin >> nomorPlat;
```

Pada saat kita ingin memasukkan motor/mobil, Pertama-tama Program akan memeriksa apakah variabel `MAKS_SLOT_MOTOR` atau yang bisa kita sebut sebagai Tempat parkir masih tersisa slot untuk parkir atau tidak,

```
if (jumlahMotor < MAKS_SLOT_MOTOR) {
```

```
    string nomorPlat;
```

Penjelasan pada Kode diatas ialah yaitu, Jika JumlahMotor (Kendaraan yang akan masuk) > (Apakah lebih Kecil) MAKS_SLOT_MOTOR (Dari Tempat Parkir) Maka Kendaraan Bisa Masuk, Lalu program akan lanjut eksekusi perintah selanjutnya yaitu:

```
    cout << "Masukkan nomor identitas motor: ";
```

```
    cin >> nomorPlat;
```

```
Pilihan: 1
Masukkan nomor identitas motor:
```

Gambar 5.10 Output 2

Program Akan Meminta Identitas Kendaraan “Masukkan nomor identitas motor:” Dan input/identitas yang kita masukkan akan di’masukkan’ kembali ke variabel nomorPlat dengan tipe data string, memakai string yaitu karena agar kita bisa memberi identitas berupa karakter lebih dari 1 Karakter Contohnya yaitu ‘Supra’

```
time_t waktuMasuk = time(0);
    nomorPlatMotor[jumlahMotor] = nomorPlat;
    waktuMasukMotor[jumlahMotor] = waktuMasuk;

    cout << "Motor berhasil masuk. Nomor tiket: " << jumlahMotor + 1 << endl;
    jumlahMotor++;
} else {
    cout << "Tempat parkir motor penuh!" << endl;
};
```

```
time_t waktuMasuk = time(0);
    nomorPlatMotor[jumlahMotor] = nomorPlat;
    waktuMasukMotor[jumlahMotor] = waktuMasuk;
```

Pada saat Kita sudah memasukkan kendaraan dan telah memberikan identitas Maka Fungsi `time_t` Akan mulai dipakai, Dengan variabel WaktuMasuk dimulai dari 0 atau waktu kendaraan kita masuk, Contohnya 12:00 PM

Dan Variabel `nomorPlat` Akan ‘Dimasukkan’ kedalam array (Pengingat: Array adalah Tipe Data yang berfungsi untuk menjadi ‘Wadah’ dari beberapa variabel dengan

tipe data yang sama) Dari `nomorPlatMotor[jumlahMotor]` Dan hal ini juga berlaku untuk variabel `waktuMasuk`, Inti dari hal yang barusan kita bahas yaitu program akan ‘memulai waktu’ dari kendaraan yang kita masukkan

```
cout << "Motor berhasil masuk. Nomor tiket: " << jumlahMotor + 1 << endl;
    jumlahMotor++;
} else {
    cout << "Tempat parkir motor penuh!" << endl;
};
```

Lalu Setelah waktu dari kendaraan yang kita masukkan sudah dimulai waktu masuknya, maka kendaraan yang kita masukkan akan diberi “Tiket” dengan diberi nilai angka, Contohnya Nomor tiket : 1

Dan program juga akan menambah nilai dari variabel `jumlahMotor` Karena sebuah “motor” Baru masuk kedalam variabel `jumlahMotor` dengan increment `jumlahMotor++`;

Contohnya yaitu jika sebelum memasuki motor variabel `jumlahMotor` bernilai 0, Dan ada motor yang akan masuk, maka program akan mengeksekusi `jumlahMotor++`;

Dan menambah karena memakai `++` maka variabel `jumlahMotor` bernilai menjadi 1

```
Pilihan: 1
Masukkan nomor identitas motor: Supra
Motor berhasil masuk. Nomor tiket: 1
```

Gambar 5.11 Output 3

Dan jika Kondisi (`jumlahMotor < MAKS_SLOT_MOTOR`) Tidak Terpenuhi Artinya yaitu `jumlahMotor` melebihi ‘tempat parkir’ maka program akan memberi tahu bahwa `"Tempat parkir motor penuh!"`

Break:

Statement **break** digunakan untuk keluar dari blok switch-case dan melanjutkan eksekusi di luar switch-case.

Dengan demikian, **case 1** ini mengatur proses parkir untuk motor, memeriksa ketersediaan slot, menyimpan informasi motor yang masuk, dan memberikan respons kepada pengguna

Dan untuk case 2 cara kerjanya sama seperti case 1 yang berbeda adalah case 2 dikhususkan untuk ‘mobil’.

5.5 Bagian [5]

```
case 3:
    int tiketKeluar;
    cout << "Pilih jenis kendaraan yang keluar:\n";
    cout << "1. Motor\n2. Mobil\nPilihan: ";
    cin >> pilihan;
```

Setelah kita membahas “Masuknya Kendaraan Pada Tempat Parkir” Dibagian ini kita akan membahas Yaitu saat kita ingin mengeluarkan kendaraan yang tadinya sudah kita masukkan, Mari kita bahas sebagai berikut:

```
case 3:
    int tiketKeluar;
    cout << "Pilih jenis kendaraan yang keluar:\n";
    cout << "1. Motor\n2. Mobil\nPilihan: ";
    cin >> pilihan;
```

Pada saat kita memilih opsi ke 3, Artinya kita akan ingin mengeluarkan kendaraan Maka kita akan diberi pilihan sebagai berikut

```
Pilihan: 3
Pilih jenis kendaraan yang keluar:
1. Motor
2. Mobil
Pilihan:
```

Gambar 5.12 Output 4

Jika kita ingin mengeluarkan Motor Maka ketik pilihan menjadi ‘1’ Dan mobil ‘2’ Maka program akan mengeksekusi perintah sebagai berikut

Lalu setelah kita memilih antara 2 pilihan tersebut, kita akan dihadapkan dengan sebuah kode yaitu dihalaman berikut:

```
if (pilihan == 1 && jumlahMotor > 0) {
    cout << "Masukkan nomor tiket motor yang keluar: ";
    cin >> tiketKeluar;
    if (tiketKeluar > 0 && tiketKeluar <= jumlahMotor) {
        string plat = nomorPlatMotor[tiketKeluar - 1];
        time_t waktuKeluar = time(0);

int
waktuTotal=static_cast<int>(difftime(waktuKeluar,waktuMasukMotor[tiketKeluar -
1]) / 60.0);

        cout << "Waktu masuk: " << ctime(&waktuMasukMotor[tiketKeluar - 1]);
        cout << "Waktu keluar: " << ctime(&waktuKeluar);
        cout << "Waktu parkir motor: " << waktuTotal << " menit" << endl;

        int biayaTotal = TARIF_JAM_PERTAMA_MOBIL;
        waktuTotal -= 60;
        int tarifJamBerikutnya = TARIF_JAM_PERTAMA_MOBIL;
        while (waktuTotal > 0) {
            biayaTotal += tarifJamBerikutnya;
            waktuTotal -= 60;
        }
        cout << "Biaya parkir motor: Rp " << biayaTotal << endl;

        for (int i = tiketKeluar - 1; i < jumlahMotor - 1; ++i) {
            nomorPlatMotor[i] = nomorPlatMotor[i + 1];
            waktuMasukMotor[i] = waktuMasukMotor[i + 1];
            waktuParkirMotor[i] = waktuParkirMotor[i + 1];
        }
        jumlahMotor--;
    } else {
        cout << "Nomor tiket tidak ditemukan!" << endl;
    }
}
```

Berikut Penjelasan langkah-langkahnya:

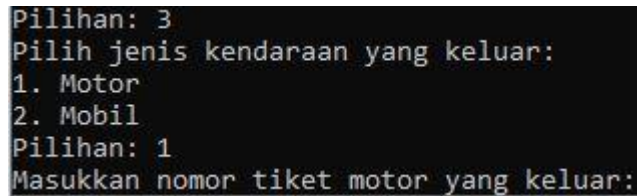
```
if (pilihan == 1 && jumlahMotor > 0) {
    cout << "Masukkan nomor tiket motor yang keluar: ";
    cin >> tiketKeluar;
    if (tiketKeluar > 0 && tiketKeluar <= jumlahMotor) {
        string plat = nomorPlatMotor[tiketKeluar - 1];
        time_t waktuKeluar = time(0);
```

Mari kita mulai analisisnya yaitu sebagai berikut:

```
if (pilihan == 1 && jumlahMotor > 0) {
    cout << "Masukkan nomor tiket motor yang keluar: ";
```

```
cin >> tiketKeluar;
```

Pada Saat kita memilih Contohnya Memilih ingin mengeluarkan “Motor”/opsi ke 1 Maka akan diperiksa dahulu beberapa kondisi dengan `&&` yaitu berfungsi untuk memeriksa, Apabila 2 kondisi benar, Maka program akan lanjut mengeksekusi Dan meminta Nomor ‘Tiket’ yang diberikan saat kita memasukkan kendaraan pada pilihan 1/2 diawal



```
Pilihan: 3
Pilih jenis kendaraan yang keluar:
1. Motor
2. Mobil
Pilihan: 1
Masukkan nomor tiket motor yang keluar:
```

Gambar 5.13 Output 5

```
if (tiketKeluar > 0 && tiketKeluar <= jumlahMotor) {
```

```
    string plat = nomorPlatMotor[tiketKeluar - 1];
```

```
    time_t waktuKeluar = time(0);
```

Setelah kita memberi nomor tiket pada kendaraan yang ingin kita keluarkan, selanjutnya program akan mengeksekusi Kode diatas, Penjelasannya yaitu

```
if (tiketKeluar > 0 && tiketKeluar <= jumlahMotor) {
```

“Jika Kedua kondisi berikut benar, maka lanjut eksekusi program, Kondisinya ialah `tiketKeluar` (Variabel yang dibuat untuk Menampung Nomor Tiket kendaraan yang ingin keluar) > Lebih kecil dari 0 `&&` (AND) `tiketKeluar` <= (Lebih kecil Sama dengan) `jumlahMotor` (Jumlah kendaraan yang sudah dimasukkan ke tempat parkir (MAKS_SLOT_MOTOR)) Maka Program akan mengeksekusi

```
string plat = nomorPlatMotor[tiketKeluar - 1];
```

Yaitu identitas/tiket yang sudah diberi oleh program pada saat kita memasuki kendaraan, Contohnya Pada saat kita masuk, kita mendapat Nomor tiket : 1

```
time_t waktuKeluar = time(0);
```

`time_t` diatas berfungsi untuk mencatat waktu keluar dari kendaraan yang kita masukkan, Contohnya bila tadi kita memasukkan kendaraan dan tercatat masuk pada jam 09:00 AM setelah itu kita ingin mengeluarkan kendaraan pada 11:00 AM maka 01:00 lah yang akan dicatat oleh variabel `waktuKeluar`

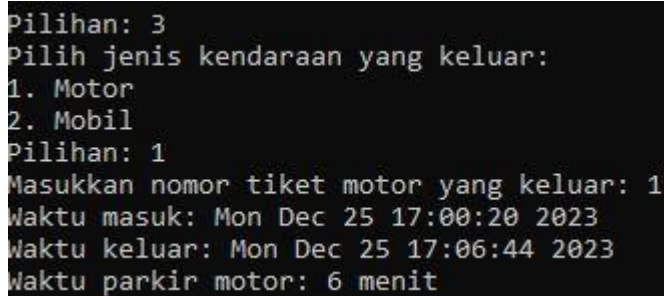
Int

```
waktuTotal=static_cast<int>(difftime(waktuKeluar,waktuMasukMotor[tiketKeluar - 1]) / 60.0);
```

Kode Diatas berfungsi untuk ‘Mentotalkan’ Waktu masuk dan Waktu Keluar dari kendaraan menggunakan salah satu fungsi yang bisa dipakai bila kita menggunakan perpustakaan `#include <ctime>` Yaitu fungsi `difftime` serta memakai `static_cast<int>` yang berfungsi untuk mengubah tipe data suatu variabel, dan untuk kasus kali ini mengubah output waktu kelaaur yang contohnya dari 0.6777 (Karena menggunakan Tipe Data bilangan real) dijadikan menjadi 6 (karena dijadikan menjadi int/integer) dan dibagi 60.0 agar menjadi “Menit”

```
cout << "Waktu masuk: " << ctime(&waktuMasukMotor[tiketKeluar - 1]);  
cout << "Waktu keluar: " << ctime(&waktuKeluar);  
cout << "Waktu parkir motor: " << waktuTotal << " menit" << endl
```

Setelah program mengeksekusi Kode diatas, maka kita akan berikan output(keluaran) yaitu:



```
Pilihan: 3  
Pilih jenis kendaraan yang keluar:  
1. Motor  
2. Mobil  
Pilihan: 1  
Masukkan nomor tiket motor yang keluar: 1  
Waktu masuk: Mon Dec 25 17:00:20 2023  
Waktu keluar: Mon Dec 25 17:06:44 2023  
Waktu parkir motor: 6 menit
```

Gambar 5.14 Output 6

Setelah kita membuat cara bagaimana program mengeluarkan waktu, maka selanjutnya yaitu biaya parkir

```
int biayaTotal = TARIF_JAM_PERTAMA_MOTOR;  
waktuTotal -= 60;  
int tarifJamBerikutnya = TARIF_JAM_PERTAMA_MOTOR;  
while (waktuTotal > 0) {  
    biayaTotal += tarifJamBerikutnya;  
}
```

```
int biayaTotal = TARIF_JAM_PERTAMA_MOTOR;  
waktuTotal -= 60;
```

`int biayaTotal = TARIF_JAM_PERTAMA_MOTOR;` Kode Disamping bertujuan untuk menetapkan terlebih dahulu Nilai `biayaTotal`, Kita membuat dahulu variabel

baru yang bernama `biayaTotal` lalu nilainya dijadikan bernilai variabel `TARIF_JAM_PERTAMA_MOTOR`, Masih ingat diawal Yaitu Variabel `TARIF` Untuk Motor Bernilai 2000.

`waktuTotal -= 60;` Kode disamping berfungsi untuk “Memotong” variabel `waktuTotal` dengan operator penugasan `-=` sebanyak 1 jam/60 menit(Karena sudah dikonversi menjadi menit pada kode diatas), Cara kerja kode disamping yaitu `waktuTotal = waktuTotal - 60`.

```
int tarifJamBerikutnya = TARIF_JAM_PERTAMA_MOTOR;
while (waktuTotal > 0) {
    biayaTotal += tarifJamBerikutnya;
    waktuTotal -= 60;
}
```

`int tarifJamBerikutnya = TARIF_JAM_PERTAMA_MOTOR;` Kode disamping sama tujuannya dengan kode diatas, yaitu menetap nilai kepada variabel yang baru juga dibuat yaitu variabel `tarifJamBerikutnya` menjadi bernilai Rp 2000. Lalu kita beri kondisi untuk memeriksa apakah waktu parkir sebuah motor sudah lewat melebihi 1 jam atau belum, yaitu dengan `while (waktuTotal > 0)` yang berfungsi untuk memeriksa apakah `waktuTotal` bernilai lebih dari 0, (0 adalah angka yang lebih kecil dari positif(1) dan lebih kecil dari negatif (-1)), jika kondisi benar maka kode berikutnya yaitu `biayaTotal += tarifJamBerikutnya;` akan dijalankan, kode tersebut berfungsi untuk “Menambah” biaya parkir,Lalu `waktuTotal -= 60;` Akan mengurangi waktu sebanyak 1jam, sampai waktu parkir kendaraan tersebut lebih kecil dari 0.

Agar lebih jelas maka kita akan membahas cara kerjanya sebagai berikut:

“Ketika sebuah motor parkir lebih dari 60 menit, Misal motor tersebut parkir selama 67menit, Maka kode diatas akan “Memotong” waktu dengan fungsi penugasan `-=` selama 1 jam, Bila 67 menit dipotong sebanyak 60 menit maka tersisa 7 menit yang berarti bernilai “lebih dari 0” maka kode berikutnya yaitu `while (waktuTotal > 0)` mengeksekusi kode berikutnya yaitu `biayaTotal += tarifJamBerikutnya;` yang akan membuat `biayaTotal` dari motor yang parkir menjadi Rp 4000 dan kode `waktuTotal -= 60;` Akan terus mengurangi waktu parkir motor tersebut sampai menjadi negatif (-).

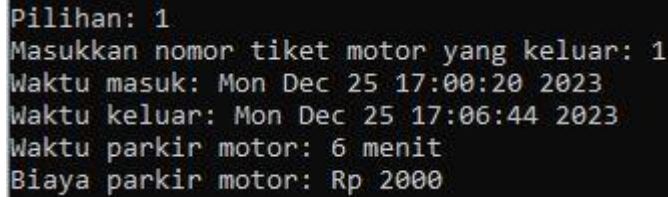
Dan jika sebuah motor parkir dibawah 60 menit maka itu berarti variabel `waktuTotal` bernilai negatif(-) yaitu kurang dari 0 dan program tidak akan

UNIVERSITAS TEKNOKRAT INDONESIA

KAMPUSNYA SANG JUARA

mengeksekusi kondisi **while** (`waktuTotal > 0`), maka biaya parkir dari motor tersebut hanya tetap menjadi Rp 2000.

Berikut adalah hasil program tersebut



```
Pilihan: 1
Masukkan nomor tiket motor yang keluar: 1
Waktu masuk: Mon Dec 25 17:00:20 2023
Waktu keluar: Mon Dec 25 17:06:44 2023
Waktu parkir motor: 6 menit
Biaya parkir motor: Rp 2000
```

Gambar 5.15 Output 7

```
for (int i = tiketKeluar - 1; i < jumlahMotor - 1; ++i) {
    nomorPlatMotor[i] = nomorPlatMotor[i + 1];
    waktuMasukMotor[i] = waktuMasukMotor[i + 1];
    waktuParkirMotor[i] = waktuParkirMotor[i + 1];
}
```

Prosesnya dapat dijelaskan sebagai berikut:

❖ Loop untuk Pemindahan Data:

Program menggunakan loop for untuk berjalan melalui data motor yang masih ada di tempat parkir.

Data motor di paling depan (dengan nomor tiket `tiketKeluar`) akan dihapus karena motor tersebut sudah keluar.

❖ Pemindahan Data:

Nomor plat, waktu masuk, dan waktu parkir motor yang ada pada posisi tertentu (indeks `i`) digantikan dengan data motor di posisi berikutnya (indeks `i + 1`).

Hal ini terjadi untuk semua motor setelah motor yang keluar.

❖ Penghapusan Data Terakhir:

Data motor terakhir di tempat parkir akan kosong atau diisi dengan data dari kendaraan lain yang keluar jika ada.

Dengan kata lain, loop tersebut menjalankan proses penggeseran data ke depan untuk mengisi tempat yang ditinggalkan oleh motor yang keluar. Setiap motor di tempat parkir digantikan oleh motor di belakangnya, dan data terakhir dihapus karena motor tersebut sudah keluar.

Misalnya seperti, jika kita sudah mengisi tempat parkir mobil dan menjadi full/15 mobil Artinya sudah ada 15 tiket, Jika mobil dengan tiket nomor 7 ingin keluar maka otomatis akan tersisa 14/15 mobil, Nah potongan kode 'for' diatas ini berfungsi

untuk ‘Menggeser’ data ke depan, Jadi, Jika kita memasuki mobil, kita tidak akan mendapat Tiket Nomor 7, Melainkan akan mendapatkan tiket nomor 15 karena sudah terjadi penggeseran

```
        jumlahMotor--;  
    } else {  
        cout << "Nomor tiket tidak ditemukan!" << endl;  
    }  
    jumlahMotor--;
```

Increment diatas berfungsi untuk mengurangi ‘nilai’ dari jumlahMotor, dengan kata lain, ketika motor keluar maka jumlah motor yang parkir pada ‘MAKS_SLOT_MOTOR’(Kapasitas/tempat parkir motor) Akan berkurang

```
    } else {  
        cout << "Nomor tiket tidak ditemukan!" << endl;  
    }
```

Dan jika kita memasuki nomor tiket yang salah/tidak ada di array/tempat parkir, maka program akan memberitahu bahwa tiket yang kita beri tidak ditemukan ditempat parkir

5.6 Bagian [6]

```
case 4:  
    cout << "\nInfo Parkir Motor:\n";  
    for (int i = 0; i < jumlahMotor; ++i) {  
        cout << "Nomor Tiket: " << i + 1 << endl;  
        cout << "Nomor Identitas: " << nomorPlatMotor[i] << endl;  
        cout << "Waktu Masuk: " << ctime(&waktuMasukMotor[i]);  
  
        time_t waktuSekarang = time(0); // Waktu saat ini  
        int waktuParkir = static_cast<int>(difftime(waktuSekarang, waktuMasukMotor[i]) / 60.0);  
  
        cout << "Waktu Parkir: " << waktuParkir << " menit" << endl;  
        cout << "-----\n";  
    }
```

Penjelasan langkah-langkahnya:

- ❖ “cout << “\nInfo Parkir Motor:\n”;
: Program Mencetak judul informasi dari case 4 yaitu parkir motor.
- ❖ “for (int i = 0; i < jumlahMotor; ++i) {
: Pengulangan diatas yaitu untuk menampilkan informasi setiap kendaraan motor yang terparkir,
- ❖ “cout << “Nomor Tiket: ” << i + 1 << endl;”
: Menampilkan nomor tiket kendaraan motor yang sudah terparkir.
- ❖ “cout << “Nomor Identitas: ” << nomorPlatMotor[i] << endl;”

: Menampilkan nomor identitas (nomor plat) kendaraan yang kita dapat pada saat ingin memasukkan kendaraan pada case 1 (motor) maupun 2 (mobil).

❖ `cout << "Waktu Masuk: " << ctime(&waktuMasukMotor[i]);`

: Menampilkan waktu masuk kendaraan .

❖ `time_t waktuSekarang = time(0);`

`int waktuParkir=static_cast<int>(difftime(waktuSekarang,waktuMasukMotor[i]) / 60.0);`

Potongan kode diatas berfungsi untuk Mentotalkan lama parkir/waktuParkir, waktu ditentukan hanya untuk kendaraan yang ada di dalam tempat parkir, jadi tidak berlaku pada kendaraan yang sudah keluar, Potongan kode diatas akan mentotalkan waktu masuk hingga Waktu sekarang/Saat ini, Misal kendaraan sudah masuk sejak 11:00AM dan saat ini adalah 02:00PM Maka Waktu parkir dari kendaraan tersebut adalah 180Menit/3jam.

disamping adalah tampilan detail informasi parkir motor yang disajikan kepada pengguna ketika mereka memilih opsi "Info Parkir" (case 4).

```
Info Parkir Motor:
Nomor Tiket: 1
Nomor Identitas: Supra
Waktu Masuk: Mon Dec 25 19:20:10 2023
Waktu Parkir: 19 menit
-----
Info Parkir Mobil:
Nomor Tiket: 1
Nomor Identitas: Avanza
Waktu Masuk: Mon Dec 25 19:23:25 2023
Waktu Parkir: 16 menit
-----
```

Gambar 5.16 Output 8

Dengan demikian, **case 4** bertanggung jawab untuk menampilkan informasi parkir baik untuk kendaraan motor maupun mobil. Dari nomor tiket, Identitas kendaraan tersebut, Waktu masuk, hingga berapa lama kendaraan tersebut sudah parkir di tempat parkir

```
default:
    cout << "Pilihan tidak valid." << endl;
    break;
}
```

Kode “default: berfungsi untuk memberi pernyataan “Pilihan tidak valid” ketika kita tidak memilih pilihan 1-4 yang ada pada program

5.7 Bagian [7]

```
cout << "\n+-----+\n";  
cout << "| Jumlah kendaraan yang masuk|\n";  
cout << "| Motor: " << jumlahMotor << " dari " << MAKS_SLOT_MOTOR << " |\n";  
cout << "| Mobil: " << jumlahMobil << " dari " << MAKS_SLOT_MOBIL << " |\n";  
cout << "+-----+\n";
```

Berfungsi untuk menampilkan jumlah kendaraan yang masuk

```
+-----+  
a Jumlah kendaraan yang masuk: a  
a Motor: 2 dari 30 a  
a Mobil: 3 dari 15 a  
+-----+
```

Gambar 5.17 Output 9

BAB 6

KESIMPULAN

6.1 Kesimpulan

Buku Ajar : Membuka Dunia Kode, Buku saku ini bertujuan untuk memberi/sharing pengetahuan Penulis mengenai dasar dasar pemrograman, Buku saku ini memiliki 6 bab, Kita memulai pelajaran dengan dasar dari pemrograman, yaitu Algoritma, kita memahami dahulu konsep dari JUDUL ALGORITMA, Yang berarti Masalah.

DEKLARASI , Yang Bisa juga dimaknakan sebagai Alat/Kebutuhan apa saja yang kita butuhkan untuk menyelesaikan masalah tersebut, Lalu

DESKRIPSI, Yaitu Aksi kita dalam melaksanakan/Mencari solusi dari masalah yang kita hadapi,

Lalu kita juga belajar sedikit tentang apa itu pseudocode dan juga flowchart, Kedua hal tersebut sangat berguna agar kita mengerti atau pada saat ingin mengajarkan pemrograman kepada seseorang, agar lebih mudah kita mengajarkannya kita lebih baik menggunakan Bahasa yang dapat dimengerti oleh orang lain, dan pseudocode maupun flowchart adalah salah satu caranya

Dan setelah kita belajar tentang Dasar dari program, Lalu kita mencoba membuka dan belajar hal baru yaitu 'Alat' yang bisa kita gunakan untuk membuat suatu program, Dibuku ini kita belajar tentang C++, 'alat' Seorang programmer bisa sangat beragam, ada yang memakai lebih dari satu 'Alat' atau juga bisa disebut sebagai 'Bahasa Pemrograman' Contohnya ialah Python, Javascript dan Lainnya, Ada juga seorang programmer yang Fokus memakai 1 bahasa pemrograman saja karena programmer tersebut sudah 'nyaman' memakai bahasa tersebut, Pada BAB 2 Kita belajar mengenai fitur fitur Tentang Aplikasi Yaitu Dev C++, Aplikasi yang mengizinkan kita untuk menggunakan bahasa pemrograman C++, Setelah kita mengetahui hal tersebut lalu kita juga belajar lagi mengenai operator Pada BAB 3, Operator pada pemrograman Hampir mirip dengan operator yang kita pelajari pada saat ada dikelas matematika, Seperti (+), (-), (/) dan lainnya.

Dan Pada BAB 4 Barulah kita Belajar mengenai 'Fungsi' Yang beragam, Dari if yang mengizinkan kita untuk membuat kondisi, For yang mengizinkan kita untuk membuat sebuah loop/Perulangan pada suatu program, Dan array yang bisa

mengizinkan kita untuk Membuat suatu ‘Tempat’ Atau ‘Wadah’ yang mengizinkan untuk lebih dari satu variabel ‘singgah’ didalam array tersebut, Tidak lupa juga kita membuka dunia kode dimulai dengan ‘Kamus’ yang membuat kita bisa menggunakan Fungsi Fungsi seperti <ctime> yang mengizinkan kita bisa menggunakan Waktu, <iostream> yang membuat program bisa ‘Mengeluarkan’ dan ‘mendengarkan’ kode yang kita berikan.

Lalu setelah kita belajar Dasar dari pemrograman, aplikasi yang mengizinkan kita bisa membuat kode, Serta Belajar mengenai Fungsi Fungsi apa saja yang bisa kita pakai pada program C++, Kita langsung lanjut membuat langsung program Yang menjadi tujuan pembelajaran kita yaitu, Program parkir, Dikhususkan untuk BAB 5 Kita akan menganalisis program yang sudah dibuat, Mulai dari ‘Kendaraan; Masuk kedalam ‘Tempat Parkir’, Hingga keluar dari ‘tempat parkir’ Ada juga opsi agar kita bisa memeriksa sudah berapa lama kita parkir di ‘tempat parkir’ tersebut, Dan Setelah itu, kita sampai pada penghujung BAB pada buku ajar ini,

6.2 Ucapan Terima Kasih Dan Penutup

Penulis sangat berterima kasih kepada Teman-teman yang telah berjuang membantu dalam pembuatan buku saku ini, Dari Dosen kami yaitu Miss Lili andraini, S.T. Yang telah menjadi pembimbing kami dalam pembuatan buku ajar ini, Juga teman teman penyusun buku ini yang telah bekerja keras dan meluangkan waktunya untuk melancarkan pembuatan Buku Ajar : Membuka Dunia Kode,

Dan penulis berterima kasih kepada pembaca karena sudah ‘membaca’ Buku Saku ini dan sekaligus minta maaf Jika ada penjelasan yang kurang dapat dimengerti oleh pembaca, Semoga ilmu yang ada pada buku ini dapat membantu pembaca dalam mendapatkan ilmu terkait pemrograman, Dapat membantu pembaca dalam belajar dasar dasar pemrograman, Sekali lagi penulis minta maaf bila Ada banyak penulisan yang Berantakan, Kurang dapat dimengerti, Rumit dimengerti, Dan Lainnya, Semoga pembaca bisa memaklumi kesalahan kesalahan yang ada pada buku ini, Dan saya sebagai penulis Sadar sekali bahwa masih banyak kekurangan dalam buku ini.

Sekian ucapan terima kasih (Dan minta Maaf) Dari penulis kepada pembaca.



BUKU AJAR

MEMBUKA DUNIA KODE

Belajar Dasar-Dasar dari Pemograman



**UNIVERSITAS
TEKNOKRAT
INDONESIA**

Jl. ZA. Pagar Alam No.9 -11, Labuhan
Ratu, Kec. Kedaton, Kota Bandar
Lampung, Lampung 35132