

Bab 7

Algoritma dan Pemrograman



Tujuan Pembelajaran

Setelah mempelajari bab ini, kalian mampu (a) membaca dan menulis algoritma dengan notasi yang benar, memahami proses pemrograman dengan menggunakan bahasa pemrograman; (b) memahami konsep variabel dan ekspresi dalam membuat program; (c) memahami penggunaan struktur kontrol keputusan, struktur kontrol perulangan, dan fungsi dalam membuat program.

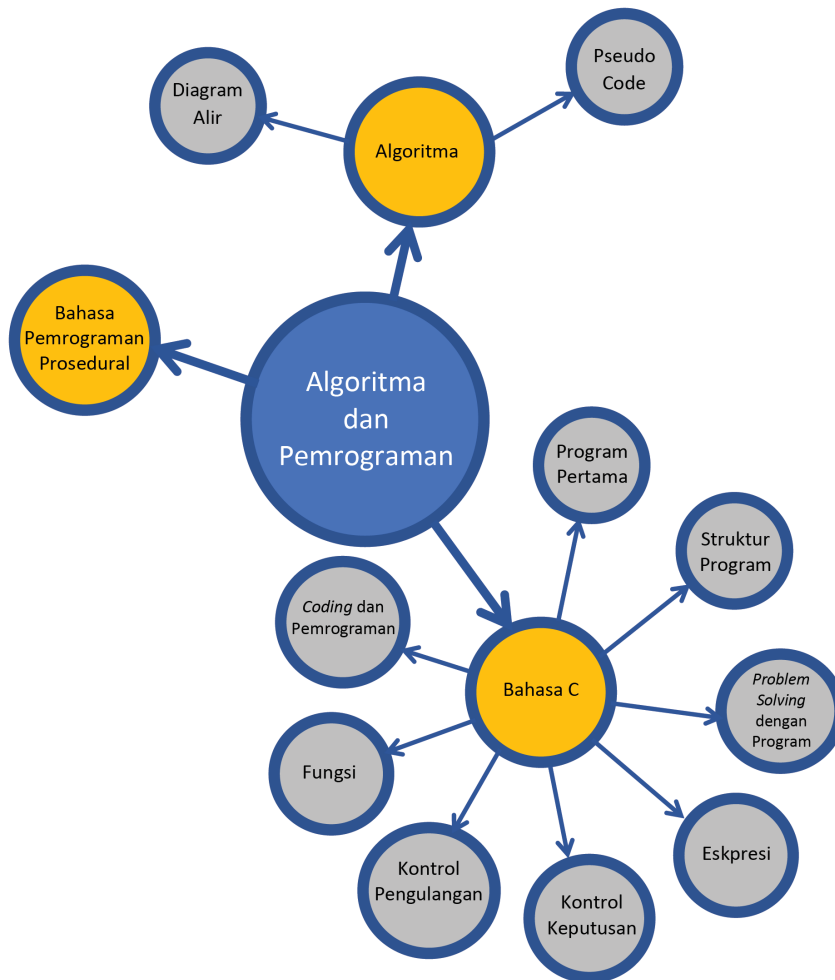


Pertanyaan Pemantik

Kalian ingin dapat memprogram dengan bahasa pemrograman? Bagaimana membuat program untuk membuat solusi-solusi kecil?



Peta Konsep



Gambar 7.1 Peta Konsep Algoritma dan Pemrograman

Apersepsi

Selama ini, mungkin kalian telah menggunakan banyak sekali produk perangkat lunak baik di komputer personal maupun di ponsel pintar yang kalian gunakan. Sekarang saatnya bagi kalian untuk membuat program sendiri. Pada bagian ini, kalian akan berkenalan dengan konsep algoritma dan pemrograman. Untuk membuat program, kalian harus menggunakan salah satu dari banyak bahasa pemrograman yang ada.



Penyelesaian Persoalan (*problem solving*), Algoritma, pemrograman, *coding*, *debugging*.

A. Algoritma

Algoritma adalah suatu kumpulan instruksi terstruktur dan terbatas yang dapat diimplementasikan dalam bentuk program komputer untuk menyelesaikan suatu permasalahan komputasi tertentu. Algoritma merupakan bentuk dari suatu strategi atau ‘resep’ yang kalian gunakan untuk menyelesaikan suatu masalah. Algoritma lahir dari suatu proses berpikir komputasional oleh seseorang untuk menemukan solusi dari suatu permasalahan yang diberikan. Dengan demikian, berpikir komputasional merupakan keahlian yang kalian perlukan untuk dapat membuat algoritma, program, atau suatu karya informatika yang dapat digunakan dengan efektif dan efisien.




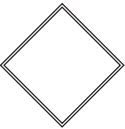



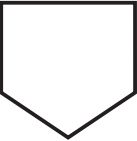
Setelah kalian menganalisis suatu problem menggunakan teknik abstraksi dan dekomposisi lalu menyusun algoritma dengan melakukan pengenalan pola dari problem sejenis, algoritma tersebut harus direpresentasikan dalam bentuk yang dapat dipahami oleh orang lain. Selain itu, karena pada akhirnya strategi tersebut akan diubah dalam bentuk kode program, algoritma harus ditulis dalam bentuk yang terdefinisi dengan baik (*well-defined*) dengan jumlah langkah yang terbatas. Algoritma adalah abstraksi dari sebuah program sehingga kemampuan menuliskan algoritma dengan baik akan membantu kalian dalam membuat program yang baik dan benar.

Pada bagian ini, kalian akan mempelajari dua cara untuk merepresentasikan algoritma, yaitu diagram alir dan *pseudocode*. Untuk itu, kalian perlu mempelajari teknik untuk membaca suatu algoritma (yang disebut penelusuran atau *tracing*) dan cara untuk menuliskan suatu algoritma. Perlu diingat bahwa menulis algoritma berbeda dengan menulis program. Program ditulis agar dapat dipahami oleh mesin, sedangkan algoritma ditulis agar dapat dipahami oleh manusia. Untuk program yang sederhana, algoritma akan sangat mirip, bahkan sama dengan program. Jika persoalan makin kompleks, algoritma hanya berisi abstraksi, yang akan mempermudah implementasinya menjadi program.

1. Diagram Alir

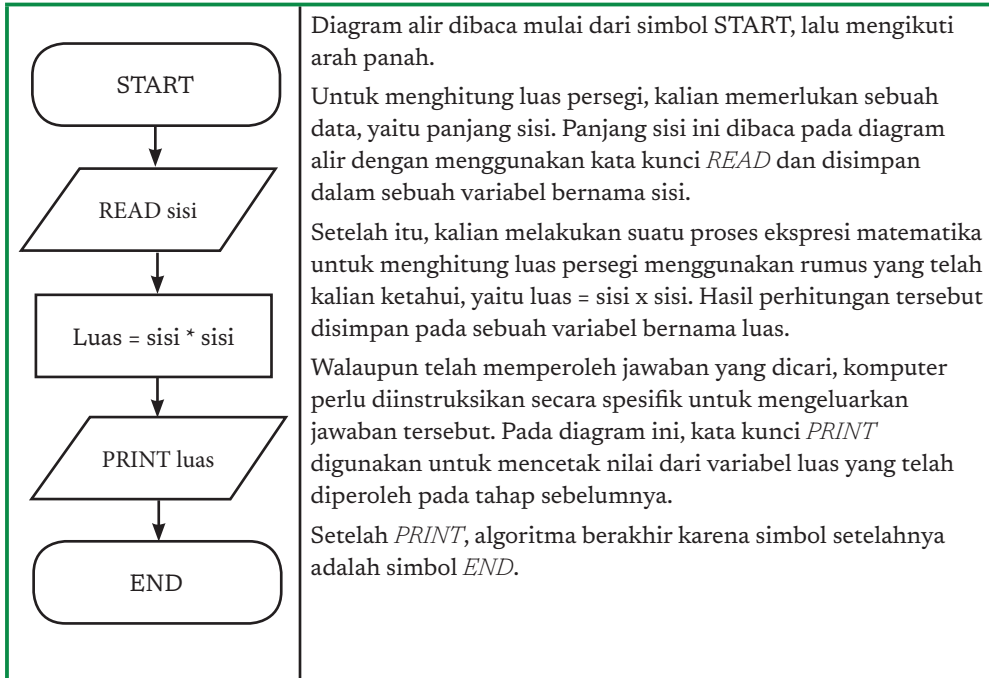
Diagram alir dibuat dalam bentuk aliran simbol yang dapat ditelusuri dari suatu titik permulaan hingga titik akhir dari program. Diagram alir dibuat menggunakan simbol standar ANSI/ISO yang beberapa simbol dasarnya diberikan pada Tabel 7.1.

Tabel 7.1 Simbol Diagram Alir Beserta Maknanya

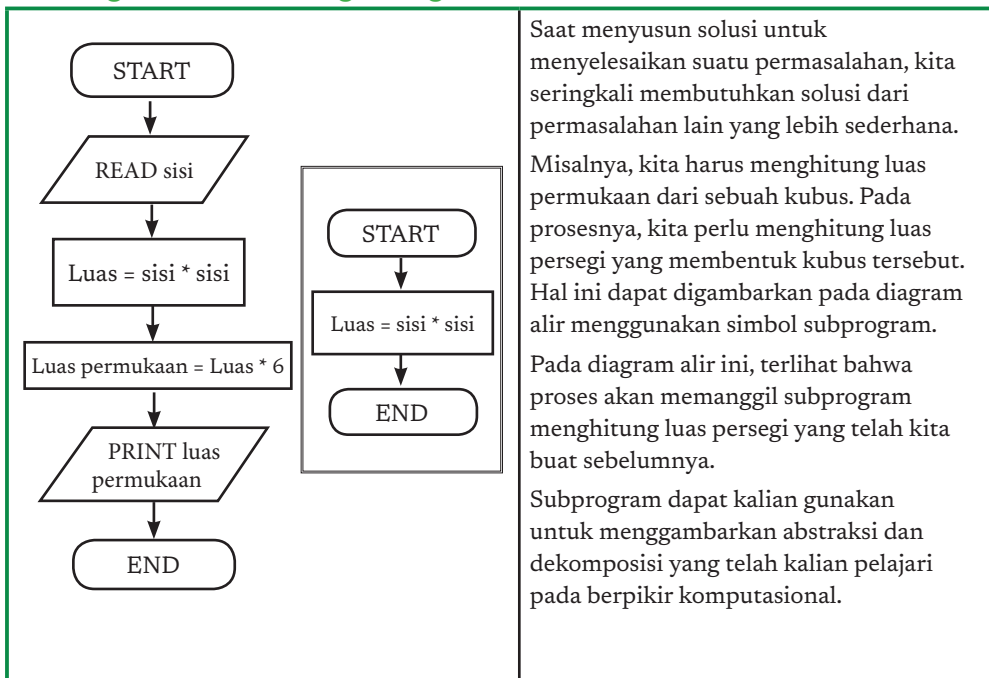
Simbol	Nama	Deskripsi
	Garis alir (<i>flowline</i>)	Arah yang menunjukkan aliran program dari awal hingga akhir.
	Terminator	Titik awal atau titik akhir suatu program.
	Proses	Suatu kegiatan komputasi yang dilakukan oleh program: misalnya operasi aritmatika.
	Keputusan	Merupakan titik percabangan yang salah satu cabangnya dapat dilalui oleh program berdasarkan suatu kondisi.
	Masukan (<i>Input</i>)/Keluaran (<i>Output</i>)	Melambangkan titik saat program akan menerima suatu data atau menghasilkan suatu informasi.
	Subprogram	Melambangkan suatu kegiatan atau proses lain yang telah didefinisikan sebelumnya.
	Penghubung dalam Halaman	Digunakan untuk menghubungkan suatu titik pada diagram alir ke titik lain pada halaman yang sama.
	Penghubung antarhalaman	Digunakan untuk menghubungkan suatu titik pada diagram alir ke titik lain pada halaman yang berbeda. Digunakan apabila diagram lain cukup kompleks sehingga tidak dapat digambar dalam satu halaman.

Untuk memahami bagaimana diagram alir digunakan untuk menggambarkan suatu algoritma, pada bagian berikut, diberikan lima buah contoh diagram alir dari beberapa proses berpikir yang telah kalian kenal.

a. Diagram Alir 1: Menghitung Luas Persegi



b. Diagram Alir 2: Menghitung Luas Permukaan Kubus



c. Diagram Alir 3: Membagi Bilangan

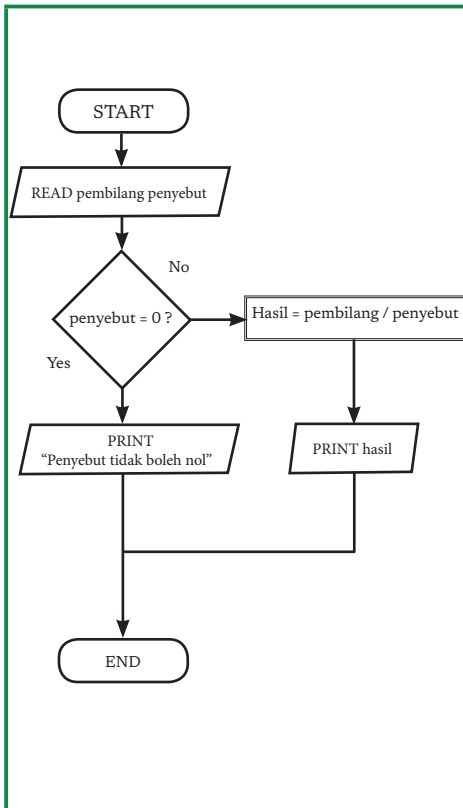
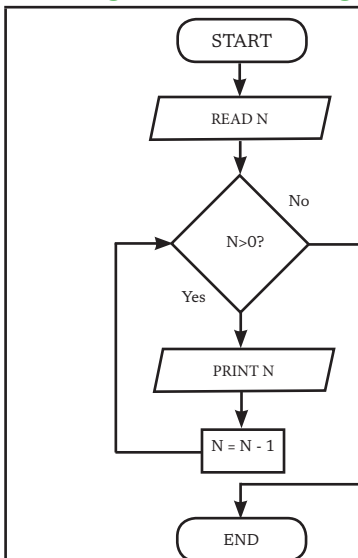


Diagram alir dapat memiliki beberapa kemungkinan aliran sehingga suatu algoritma dapat adaptif terhadap masukan yang diberikan. Hal ini dimungkinkan dengan adanya simbol keputusan. Aliran keluar dari simbol keputusan akan bergantung pada kondisi yang ada di dalam simbol keputusan.

Pada contoh ini, simbol keputusan digunakan untuk menghindari dijalankannya suatu operasi matematika yang tidak dapat dieksekusi oleh komputer, yaitu operasi pembagian dengan pembagi bernilai 0. Apabila operasi tersebut dilakukan, komputer akan menampilkan pesan kesalahan dan program akan berhenti secara tidak wajar.

Diagram alir ini merupakan proses untuk membagi pembilang dengan penyebut. Akan tetapi, sebelum operasi pembagian dilakukan, diagram akan mengecek terlebih dahulu nilai dari penyebut. Apabila penyebut bernilai 0, operasi pembagian tidak dilakukan dan pesan yang sesuai akan ditampilkan. Jika tidak, operasi dapat dilakukan dengan aman dan hasil pembagian dapat ditampilkan.

d. Diagram Alir 4: Menghitung Mundur dari N hingga 1

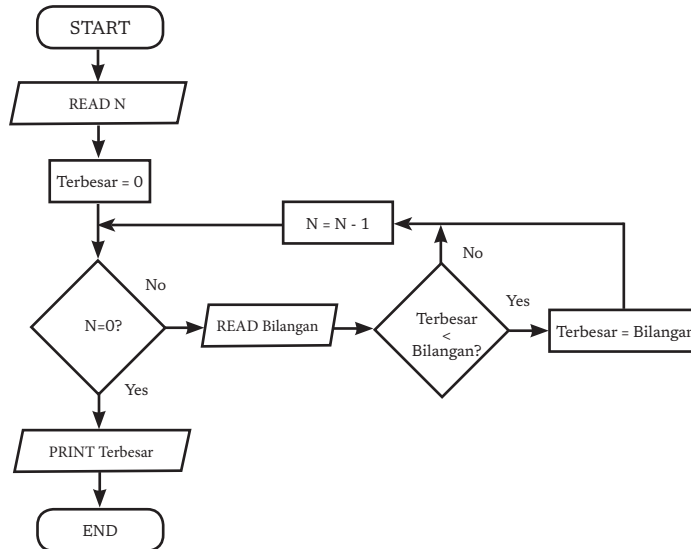


Aliran pada diagram alir dapat diatur sehingga satu lebih simbol dijalankan berulang kali. Pada contoh berikut, perulangan dilakukan sehingga diagram alir tersebut akan menghasilkan barisan bilangan bulat dari N hingga 1. Misalkan, N bernilai 5. Maka, diagram alir akan mencetak angka 5 4 3 2 1.

Tentunya, perulangan tidak bisa dilakukan terus-menerus sehingga diperlukan suatu kondisi untuk menghentikan perulangan. Simbol keputusan dapat digunakan untuk menghentikan perulangan tersebut pada kondisi yang kita tetapkan.

e. Diagram Alir 5: Mencari Bilangan Terbesar dari Suatu Himpunan Bilangan

Tentunya, simbol-simbol dasar pada diagram alir dapat dipadukan untuk menghasilkan sebuah proses yang lebih kompleks. Diagram alir berikut menggambarkan proses mencari bilangan terbesar dari suatu himpunan bilangan yang diberikan. Diagram alir berikut akan membaca sebanyak N buah bilangan dan akan menghasilkan bilangan yang paling besar di antara bilangan tersebut.



f. Menelusuri Diagram Alir

Di buku kerja kalian, kalian dapat melakukan penelusuran (*tracing*) secara terstruktur dengan membuat tabel sederhana yang terbagi tiga bagian, yaitu masukan, nilai variabel, dan keluaran. Bagian masukan akan diisi dengan data yang akan diproses, bagian nilai variabel akan menjadi tempat kalian mencatat nilai yang disimpan di dalam variabel, sedangkan bagian keluaran akan diisi dengan data yang dihasilkan oleh diagram alir. Penelusuran ini menjadi penting saat kalian ingin memahami perilaku dari suatu algoritma, atau saat kalian ingin mengecek ketepatan keluaran dari suatu algoritma.

Sebagai contoh, misal kalian mencari bilangan terbesar dari 4 bilangan berikut: 1, 3, 2, 4 menggunakan diagram alir kelima. Ada beberapa kegiatan inti yang akan kalian lakukan dalam melakukan penelusuran.

1. Mempersiapkan lembar kerja penelusuran (Tabel 7.2.a), kalian menuliskan data yang akan diolah, menuliskan nama variabel yang digunakan pada diagram alir, dan mengosongkan bagian keluaran.
2. Setelah itu, kalian mulai menelusuri diagram alir dari bagian permulaan.

3. Ketika menemukan simbol untuk membaca suatu data, kalian dapat mencoret masukan yang dibaca, kemudian meletakkannya ke variabel yang tepat. Misal, kalian berada pada simbol masukan READ N. Letakkan bilangan pertama di bagian masukan (yaitu 4) ke variabel N (Tabel 7.2.b).
4. Ketika kalian tiba di suatu simbol proses yang menyimpan suatu nilai pada variabel (penugasan atau *assignment*), kalian meletakkan nilai tersebut pada bagian nilai variabel. Misalnya, saat bertemu dengan Terbesar = 0 pada diagram alir, kalian menuliskan nilai 0 pada variabel terbesar (Tabel 7.2.c).
5. Proses juga dapat berisi ekspresi matematika, misalnya $N = N - 1$. Untuk mengerjakan ekspresi tersebut, kerjakan dahulu bagian kanan dari ekspresi, yaitu $N - 1$. Cek nilai N saat ini di bagian nilai variabel, dan kalian akan menemukan nilai 4. Kerjakan ekspresi tersebut, yaitu $4 - 1 = 3$, lalu simpan hasilnya ke sisi kanan dari ekspresi yaitu variabel N. Lewat ekspresi ini, nilai N yang tadinya 4 sekarang telah berubah menjadi 3. Kalian dapat mencoret nilai 4 dan menuliskan nilai 3 pada Lembar Kerja (Tabel 7.2.d).
6. Kemudian, kalian dapat melanjutkan proses dan akhirnya menemukan nilai N sekarang bernilai 0. Lembar Kerja kalian akan berisi seperti Tabel 7.2.e.
7. Kemudian, kalian menemukan simbol keluaran PRINT Terbesar. Pada tahap ini, kalian dapat menuliskan isi dari variabel terbesar ke dalam bagian keluaran di lembar kerja kalian (Tabel 7.2.f).
8. Terakhir, kalian menelusuri dan menemukan terminator END sehingga penelusuran berakhir. Dengan demikian, diagram alir tadi menghasilkan keluaran berupa nilai 4 pada kasus yang diberikan. Selamat! Kalian telah berhasil menelusuri diagram alir!

Tabel 7.2 Ilustrasi Penelusuran Diagram Alir di Buku Kerja Siswa.

(a) Sebelum kalian menelusuri:	(b) Saat menemukan simbol masukan dengan perintah READ N.	(c) Saat menemukan simbol proses dengan perintah terbesar = 0.
Masukan 4 1 3 2 4	Masukan 4 1 3 2 4	Masukan 4 1 3 2 4
Nilai variabel N: Bilangan: Terbesar:	Nilai variabel N: 4 Bilangan: Terbesar:	Nilai variabel N: 4 Bilangan: Terbesar: 0
Keluaran	Keluaran	Keluaran

(d) Saat menemukan simbol proses dengan ekspresi $N = N - 1$.	(e) Saat N bernilai 0, Lembar Kerja kalian akan menjadi seperti ini.	(c) Saat menemukan simbol proses dengan perintah terbesar = 0.
Masukan 4 1 3 2 4	Masukan 4 1324	Masukan 4 1324
Nilai variabel N: 4 3 Bilangan: Terbesar:	Nilai variabel N: 4321 0 Bilangan: 1324 0 Terbesar: 4	Nilai variabel N: 4321 0 Bilangan: 1324 0 Terbesar: 4
Keluaran	Keluaran	Keluaran 4

Walaupun ilustrasi ini terdiri atas beberapa tabel, kalian cukup bekerja dengan satu tabel saat menelusuri diagram alir. Teknik penelusuran ini tidak hanya dapat digunakan untuk membaca suatu diagram alir, tetapi juga dapat digunakan untuk membaca pseudocode atau kode program. Untuk algoritma yang pendek, kalian mungkin tidak membutuhkan lembar kerja seperti ini. Namun, lembar kerja ini akan sangat bermanfaat ketika menelusuri suatu algoritma yang panjang dan kompleks. Sekarang, saatnya kalian berlatih menggunakan teknik ini!



Ayo Lakukan

Aktivitas Individu

Aktivitas AP-K10-01-U: Menelusuri Diagram Alir

Sebelum mencoba membuat diagram alir sendiri, kalian perlu kemampuan untuk membaca diagram alir yang telah tersedia. Oleh karena itu, pada latihan ini, kalian akan mencoba menelusuri diagram alir yang telah tersedia di atas ketika diberikan suatu kasus untuk diselesaikan. Walaupun kalian telah mengetahui apa hasilnya, lakukanlah penelusuran secara terstruktur mulai dari awal hingga akhir diagram alir.

1. Hitunglah luas permukaan sebuah kubus yang memiliki panjang sisi 20 cm dengan menggunakan diagram alir pada contoh kedua.
2. Hitunglah dua buah operasi pembagian berikut dengan menggunakan diagram alir pada contoh kedua. Pertama, pembagi bernilai 10 dan penyebut bernilai 2. Kedua, pembagi bernilai 8 dan penyebut bernilai 0.
3. Lakukan hitung mundur dari angka 5 hingga 1 dengan menelusuri diagram alir keempat.

Carilah bilangan terbesar dari 7 bilangan berikut: 3, 1, 2, 4, 5, 7, dan 9 dengan menggunakan diagram alir kelima.

2. Pseudocode

Pseudocode (kode semu atau kode pseudo) adalah suatu bahasa buatan manusia yang sifatnya informal untuk merepresentasikan algoritma. *Pseudocode* dibuat untuk menutupi kekurangan diagram alir dalam merepresentasikan konsep-konsep pemrograman terstruktur. *Pseudocode* memungkinkan representasi langkah-langkah yang lebih detail dan dekat dengan bahasa pemrograman. Karena sifatnya yang informal, tidak ada aturan khusus dalam standar notasi yang dapat digunakan. Akan tetapi, ada beberapa prinsip dasar yang perlu diperhatikan, yaitu satu baris untuk satu pernyataan (*statement*) dan pentingnya indentasi dalam menuliskan pernyataan. Indentasi ada untuk hierarki dari pernyataan. Misalnya, untuk menunjukkan bahwa suatu pernyataan merupakan bagian dari sebuah struktur kontrol keputusan atau struktur kontrol perulangan (lihat konsep blok pada pemrograman visual yang telah kalian pelajari di tingkat SMP).

Keempat diagram alir pada bagian sebelumnya dapat ditulis dalam bentuk *pseudocode* sebagai berikut.

a. *Pseudocode* 1: Menghitung Luas Persegi

Deskripsi tingkat tinggi:	Pseudocode:
1. Baca nilai sisi persegi.	<i>Algoritma Menghitung Luas Persegi</i>
2. Hitung luas dengan mengkuadratkan nilai sisinya.	<i>Input: Nilai Panjang Sisi Persegi.</i>
3. Cetak luas.	<i>Output: Luas Persegi tercetak.</i>
	 <i>input sisi</i> <i>luas ← sisi * sisi</i> <i>print luas</i>

b. *Pseudocode* 2: Menghitung Luas Permukaan Kubus

Deskripsi tingkat tinggi:	Pseudocode:
1. Baca nilai sisi kubus.	<i>Algoritma Menghitung Luas</i>
2. Hitung luas (persegi) dari sisi kubus. Kalikan luas kubus dengan angka 6 (banyaknya jumlah persegi pada kubus) untuk mendapatkan luas permukaan.	<i>Permukaan Kubus</i>
3. Cetak luas_permukaan.	<i>Input: Nilai Panjang Sisi Kubus.</i>
	<i>Output: Luas Permukaan Kubus tercetak.</i>
	 <i>input sisi</i> <i>luas_permukaan ← luas(sisi) * 6</i> <i>print luas_permukaan</i>

c. Pseudocode 3: Membagi Bilangan

Deskripsi tingkat tinggi:	Pseudocode:
<ol style="list-style-type: none"> 1. Baca nilai pembilang dan penyebut. 2. Jika penyebut bernilai 0, cetak tulisan "Penyebut tidak boleh nol". 3. Jika penyebut tidak bernilai nol, lakukan pembagian pembilang dengan penyebut dan simpan hasilnya. 4. Cetak hasil pembagian. 	<p><i>Algoritma Membagi Bilangan</i> <i>Input: Pembilang dan Penyebut.</i> <i>Output: Hasil Pembagian tercetak.</i></p> <pre> input pembilang, penyebut if penyebut = 0 print "Penyebut tidak boleh nol" else hasil ← pembilang / penyebut print hasil </pre>

d. Pseudocode 4: Menghitung Mundur dari N hingga 1

Deskripsi tingkat tinggi:	Pseudocode:
<ol style="list-style-type: none"> 1. Baca nilai N. 2. Selama $N > 0$, ulangi. 3. Cetak tulisan N. 4. Kurangi nilai N dengan 1. 	<p><i>Algoritma Menghitung Mundur dari N hingga 1</i> <i>Input: Nilai N.</i> <i>Output: Angka hasil hitung mundur dari nilai N sampai 1 tercetak.</i></p> <pre> input N while N > 0 print N N ← N - 1 </pre>

e. Pseudocode 5: Mencari Bilangan Terbesar dari Suatu Himpunan Bilangan

Deskripsi tingkat tinggi:	Pseudocode:
<ol style="list-style-type: none"> 1. Jika himpunan bilangan kosong, maka tidak ada bilangan terbesar. 2. Jika himpunan bilangan tidak kosong, asumsikan bilangan pertama sebagai bilang terbesar saat ini. 3. Untuk setiap bilangan anggota himpunan: bandingkan bilangan tersebut dengan bilangan terbesar saat ini. Apabila bilangan tersebut lebih besar, maka bilangan tersebut akan menjadi bilangan terbesar saat ini. 	<p><i>Algoritma Mencari Bilangan Terbesar</i> <i>Input: Himpunan bilangan L.</i> <i>Output: Bilangan terbesar pada himpunan bilangan L</i></p> <pre> if size of (L) = 0 return null largest ← elemen pertama L for each item in L, do if item > largest, then largest ← item return largest </pre>

Deskripsi tingkat tinggi:	Pseudocode:
4. Apabila langkah 3 telah dilakukan pada seluruh bilangan, bilangan terbesar saat ini akan menjadi bilangan terbesar di himpunan bilangan tersebut.	

Setelah selesai menyusun suatu algoritma, barulah suatu program dibuat dengan menggunakan bahasa pemrograman tertentu. Ada banyak bahasa yang dapat digunakan, misalnya bahasa C yang digunakan pada unit ini dan bahasa Python yang digunakan pada unit analisis data.



Ayo Lakukan

Aktivitas Individu

Aktivitas AP-K10-02-U: Menulis Algoritma

Pada latihan ini, kalian diminta untuk menuliskan suatu algoritma berdasarkan deskripsi berikut. Deskripsi ini memuat narasi tingkat tinggi dari algoritma yang perlu kalian buat dalam bentuk diagram alir dan *pseudocode*. Setelah selesai, kalian dapat menunjukkan hasil pekerjaan kalian kepada teman kalian untuk ditelusuri.

Soal 1: Membayar Bakso (Tingkat Kesulitan: ★ ★)

Buatlah sebuah diagram alir atau *pseudocode* dari proses berikut.

Sebuah mesin pembayaran otomatis dirancang untuk mampu menangani pembayaran pembelian bakso secara mandiri. Mesin ini mampu untuk memberikan kembalian dalam bentuk uang kertas atau uang logam. Mesin akan menerima dua buah masukan, yaitu total bayar dan jumlah uang yang dibayarkan oleh pelanggan. Apabila jumlah uang yang dibayarkan lebih besar atau sama dengan total bayar, mesin akan menghitung kembalian yang harus diberikan kepada pelanggan. Apabila terjadi sebaliknya, mesin akan menampilkan teks “Uang yang dibayarkan kurang”.



Setelah diagram alir selesai, kalian dapat menelusurinya dengan menggunakan kasus berikut.

Kasus	Masukan	Keluaran
1	Total Bayar: 10000 Jumlah Uang: 15000	5000
2	Total Bayar: 20000 Jumlah Uang: 10000	Uang yang dibayarkan kurang.

Soal 2: Hadiah Bakso Gratis (Tingkat Kesulitan: ★ ★ ★)

Kalian adalah pengusaha bakso yang sukses. Agar usaha bakso kalian bisa lebih berkembang, kalian berencana untuk menambah sentuhan teknologi sehingga beberapa proses dapat berjalan secara otomatis. Inovasi yang kalian pikirkan ialah menggunakan sistem poin untuk memberikan diskon pada pelanggan. Poin ini akan diberikan pada saat pelanggan membayar di mesin pembayaran yang akan kalian buat. Setiap membayar, pelanggan akan menerima poin senilai harga bakso yang ia beli. Apabila total poin mencapai 100.000, pelanggan akan menerima satu porsi bakso gratis.

Kalian lalu memikirkan suatu proses berikut: setelah memesan bakso, pelanggan dapat membayar dengan menggunakan ponsel miliknya. Kemudian, mesin tersebut menambahkan total pembayaran ke total poin yang saat ini dimiliki oleh pelanggan. Apabila total poin yang dimiliki pelanggan lebih besar dari 100.000, mesin akan mengeluarkan kalimat “Anda mendapatkan kupon bakso gratis” dan mengurangi total poin pelanggan dengan nilai 100.000. Setelah itu, mesin akan menampilkan total poin pelanggan saat ini.

Setelah diagram alir selesai, kalian dapat menelusurinya dengan menggunakan kasus berikut.

Kasus	Masukan	Keluaran
1	Total Pembayaran: 80000 Total Poin Pelanggan Saat Ini: 10000	Poin Anda saat ini: 90000
2	Total Pembayaran: 20000 Total Poin Pelanggan Saat Ini: 90000	Anda mendapatkan kupon bakso gratis! Poin Anda saat ini: 10000



Ayo Kita Renungkan

Jawablah pertanyaan berikut dalam Lembar Refleksi pada Buku Kerja, dan jangan lupa mencatat kegiatan dalam Jurnal.


1. Apakah kalian merasa bahwa membuat algoritma dapat mempermudah kalian dalam menyelesaikan masalah?
2. Menurut kalian, kapan kalian akan menggunakan diagram alir dan kapan kalian akan menggunakan *pseudocode*? Mana yang paling mudah untuk kalian gunakan? Apa kelebihan dan kekurangan dari setiap pendekatan tersebut yang kalian rasakan?

3. Setelah ini, apa yang akan kalian lakukan untuk bisa membaca dan menulis algoritma dengan baik?
4. Pelajaran paling berkesan apa yang kalian dapatkan dari pertemuan ini?

B. Bahasa Pemrograman Prosedural

Belajar bahasa pemrograman sama halnya dengan belajar bahasa apa pun, dimulai dengan secara intuitif mengenal dan langsung memakai bahasa tersebut untuk keperluan sehari-hari yang penting sesuai kebutuhan, bukan dari teori bahasa. Seseorang dengan bahasa ibu bahasa Indonesia, saat belajar bahasa Inggris, akan mulai mengenal bahasa Inggris melalui “membaca” contoh-contoh kalimat sederhana yang sangat diperlukan dalam kehidupan sehari-hari, seperti mulai menyapa “Selamat pagi.”, “Jam berapa?”. Melalui contoh tersebut, ia akan belajar kosakata penting dan pola kalimat, misalnya kalimat pernyataan SPOK (Subjek, Predikat, Objek, Keterangan); struktur kalimat tanya, kalimat aklamasi, dan lain-lain. Selanjutnya, baru beranjak ke konsep yang lebih kompleks yang ada pada bahasa asing tersebut dan mengenal tata bahasa secara lebih formal dan mulai menulis.

Belajar pemrograman pada hakikatnya sama dengan belajar bahasa natural (bahasa manusia sehari-hari): seseorang belajar dari “membaca” program terlebih dulu, daripada “menulis” kode program. Proses menulis kode (*coding*) dapat dimulai setelah kalian membaca contoh-contoh program yang menjadi pola pembangun program kompleks. Bedanya dengan belajar bahasa natural, teks dalam bahasa pemrograman yang ditulis bukan dilafalkan dan dipahami sebagai teks “statis”, melainkan juga dapat dipahami oleh mesin dan dapat dieksekusi (dijalankan).



Gambar 7.2 Elemen Generik dari Bahasa Pemrograman Prosedural
Sumber: Dokumen Kemendikbud, 2021

Terdapat banyak bahasa pemrograman, dan setiap bahasa memiliki paradigma, keunggulan, tantangan masing-masing. Pada unit ini, kalian diperkenalkan pada bahasa pemrograman C yang merupakan salah satu bahasa pemrograman prosedural. Saat mempelajari bahasa C pada unit ini di kelas X, kalian akan mempelajari empat elemen generik, yaitu variabel, ekspresi, struktur kontrol keputusan, dan struktur kontrol perulangan (Gambar 7.2). Empat elemen ini berlaku di semua bahasa pemrograman prosedural lainnya. Teks kode program dalam bahasa-bahasa pemrograman lain banyak yang mirip dengan teks bahasa C.

Oleh karena itu, kalian perlu menyadari bahwa unit ini tidak dibuat hanya agar kalian menguasai pemrograman dengan bahasa C, tetapi bagaimana kalian dapat menggunakan keempat elemen dasar tersebut dalam membuat suatu program.

C. Bahasa Pemrograman C

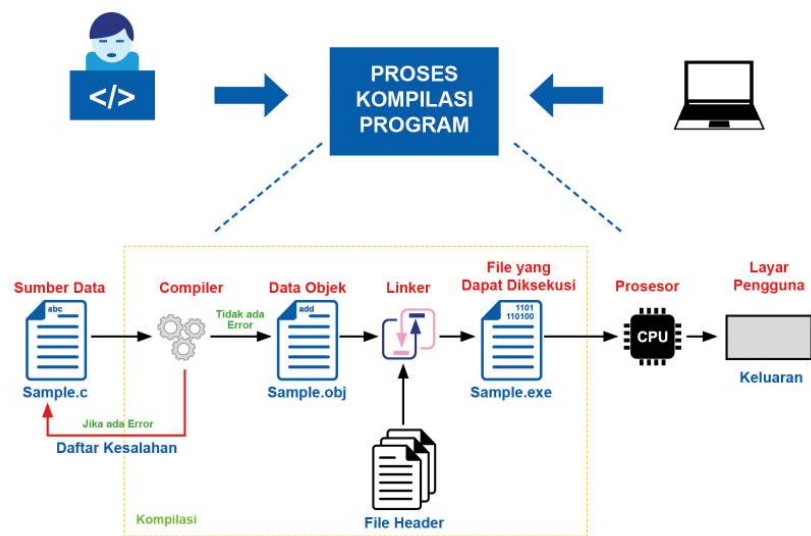
Bahasa Pemrograman C, selanjutnya disebut bahasa C saja, dikembangkan oleh Dennis M. Ritchie dan Brian W. Kernighan pada awal tahun 1970. Perkembangan bahasa C tidak bisa dipisahkan dari perkembangan sistem operasi UNIX, yang 90% lebih di antaranya ditulis dalam bahasa C. Karena sejarah yang panjang tersebut, kemudian muncul beberapa standar bahasa C yang merupakan spesifikasi dari bahasa C. Spesifikasi ini akan berpengaruh pada perilaku program yang dibuat dengan menggunakan bahasa tersebut. Beberapa standar yang ada meliputi: definisi Kernighan & Ritchie (K&R), ANSI-C (X-3.159 -1989-), Definisi AT&T (untuk superset C, C++), dan GNU Coding Standards. Versi pada PC misalnya: Lattice C, Microsoft C/Microsoft QuickC, dan Turbo C/Borland C++.

Bahasa C banyak dipakai untuk membuat sistem operasi dan program-program sistem, pemrograman tingkat rendah, atau yang "dekat" ke perangkat keras (misalnya untuk kontrol peralatan), membuat toolkit pemrograman, dan menulis aplikasi. Kelebihan bahasa C sehingga banyak digunakan ada pada kemampuannya untuk menghasilkan kode yang singkat, efisien, tetapi tetap mudah dibaca. Berbeda halnya dengan bahasa mesin yang efisien, tetapi membutuhkan latihan khusus untuk membacanya, atau bahasa tingkat tinggi lain yang enak dibaca, tetapi tidak efisien. Walaupun demikian, perlu diakui bahwa kesulitan untuk membaca program bahasa C lebih tinggi daripada bahasa tingkat tinggi lain.

1. Membuat Program Pertama dengan Bahasa C

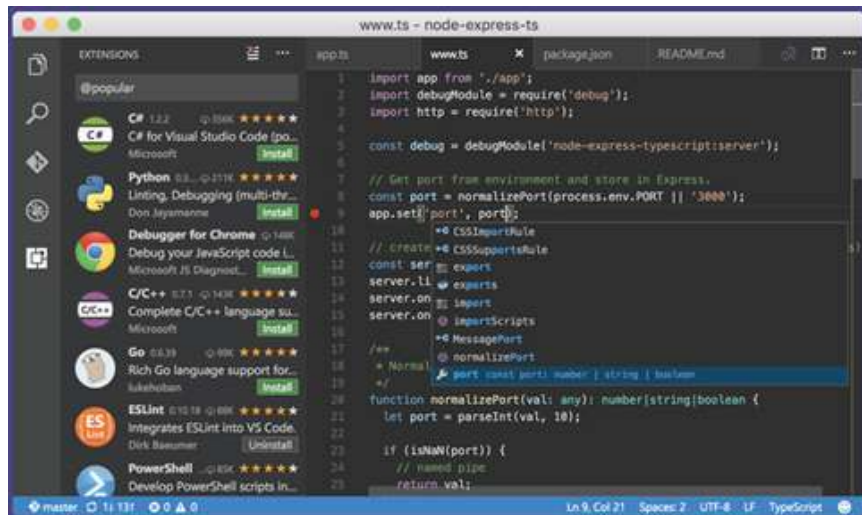
Sekarang, saatnya kalian memulai perjalanan kalian dalam membuat program dengan bahasa C. Namun, sebelum kalian mulai membuat program, ada persiapan yang harus kalian lakukan. Pertama, kalian membutuhkan sebuah tempat untuk bekerja, yang disebut lingkungan pengembangan. Kedua, kalian harus memahami proses membuat program mulai dari mengetikkan kode program hingga menghasilkan program yang dapat dieksekusi atau dijalankan oleh komputer.

a. Persiapan Lingkungan Pengembangan Program Menggunakan Bahasa C



Gambar 7.3 Alur proses membuat program dengan Bahasa C

Ini langkah-langkah untuk membuat program di C! (rumit sekali). Namun sekarang, ada IDE (*Integrated Development Environment*), semuanya jadi mudah! Tulis program, satu kali klik, langsung bisa tampil....



Gambar 7.4 Contoh tampilan sebuah IDE

Sumber: Dokumen Kemendikbud, 2021

Untuk dapat membuat program dalam bahasa C, diperlukan beberapa perangkat lunak. Perangkat lunak yang pertama ialah sebuah editor teks tempat kita mengetikkan kode program. Selanjutnya, ada sebuah kompilator (*compiler*) yang akan membaca kode bahasa C yang telah ditulis dan mengubahnya menjadi bahasa mesin, atau bahasa *assembly*. Setelah itu, terdapat sebuah assembler yang akan mengubah bahasa mesin tersebut ke dalam kode biner yang dapat dipahami dan dieksekusi oleh komputer. Terakhir, terdapat sebuah penghubung (*linker*) yang akan menyatukan beberapa berkas yang dihasilkan dalam proses-proses sebelumnya ke dalam sebuah bentuk berkas yang dapat dieksekusi (*executable*).

Pada awalnya, perangkat lunak tersebut terpisah, tetapi untuk memudahkan, akhirnya, dibuatlah sebuah perangkat lunak terintegrasi yang mencakup semua perangkat lunak di atas. Perangkat lunak tersebut disebut lingkungan pengembangan terpadu (*integrated development environment*). Untuk bahasa C, beberapa IDE yang biasa digunakan ialah Eclipse, Atom, Code::Blocks, Geany, dan Visual Studio. Walaupun pada dasarnya semua IDE tersebut memiliki fungsi yang sama, tetapi terdapat perbedaan pada fitur-fitur tambahan yang membuat proses pemrograman menjadi lebih mudah dilakukan. Misalnya, fitur untuk membuat program secara kolaboratif, integrasi dengan repositori kode program daring, serta fitur auto-complete. Selain IDE yang dipasang di komputer masing-masing, terdapat pula IDE yang terpasang di *cloud* dan dapat diakses secara daring, misalnya Ideone, tutorialspoint.com, dan onlinegdb.com. Selain itu, terdapat juga CppDroid, Mobile C, dan Coding C yang dapat digunakan pada ponsel cerdas.

Suatu perusahaan di bidang informatika biasanya telah memiliki standar masing-masing dari lingkungan pengembangan yang digunakan. Menguasai lingkungan pengembangan secara maksimal akan membuat kalian lebih produktif untuk menghasilkan program yang efisien dan berkualitas tinggi, terutama ketika kalian bekerja dalam tim.



Ayo Lakukan

Aktivitas Individu

Aktivitas AP-K10-03: Instalasi IDE Bahasa C

Sebelum membuat program, lingkungan kerja di komputer harus dipersiapkan. Pada bagian ini, akan dijelaskan cara instalasi IDE untuk memprogram bahasa C yang dapat digunakan secara gratis, yaitu Code::Blocks dan Geany. Cara instalasi yang diberikan ialah untuk dua sistem operasi, yaitu MS Windows dan Linux.

Sebelum lanjut, pastikan kalian telah mengetahui versi sistem operasi yang kalian gunakan, 32 bit atau 64 bit. Informasi ini dapat diperoleh dengan cara berikut.

1. Pada sistem operasi MS Windows: klik kanan ikon My Computer atau This PC, kemudian pilih Properties. Pada bagian System Type, terdapat informasi jumlah bit yang digunakan oleh komputer kalian.
2. Pada sistem operasi Linux: masuk ke terminal dan ketikkan perintah `uname -a`. Hal ini juga dapat dicek melalui antarmuka pengguna grafis pada bagian System Settings -> Details.



Ayo Lakukan

Aktivitas Individu

Aktivitas AP-K10-04-U: Membuat Program Pertama dengan Bahasa C

Pada aktivitas ini, kalian akan menggunakan Code::Blocks atau Geany untuk membuat program pertama kalian. Program yang akan ditulis ialah seperti berikut.

```
/* Program Pertamaku: */
/* Membuat program untuk mencetak "Halo Dunia!" */

#include<stdio.h>
int main() {
    printf("Halo Dunia!\n");
    return 0;
}
```

Ketikkanlah kode program tersebut ke dalam IDE yang kalian gunakan. Pada saat mengetik, kalian dapat mencoba memikirkan apa makna dari kode yang telah kalian ketikkan. Pastikan kalian mengetikkan semua kode dengan sempurna karena bahasa pemrograman sangat sensitif pada kesalahan pengetikan. Misalnya, ketika kalian lupa mengetikkan karakter “;” program kalian akan dianggap mengalami kesalahan sintaks oleh kompilator.

Petunjuk:

1. Pertama, buka Code::Blocks atau Geany. Kemudian, pilih File → New → Empty File.
2. Kedua, ketikkan kode yang diberikan di atas.
3. Ketiga, simpan berkas kode dengan nama `hello_world.cpp`. Simpan dalam sebuah direktori yang kalian buat di komputer. Namai berkas kode dan direktori dengan benar agar dapat kalian temukan dengan mudah nanti. Menyimpan kode dengan terstruktur merupakan salah satu praktik baik dalam menulis kode program.

4. Keempat, pilih Build → Build atau tekan Ctrl-F9 untuk melakukan kompilasi program. Pada tahap ini, kode program akan dikonversi ke dalam bentuk yang dapat dipahami dan dieksekusi oleh komputer.
5. Terakhir, pilih Build → Run atau tekan Ctrl-F10 untuk menjalankan program. Setelah dikompilasi dan dijalankan, program ini akan mencetak keluaran ke layar monitor yang berupa sebaris teks bertuliskan:

Halo Dunia!

Selamat, kalian telah berhasil menjalankan program pertama kalian.

Praktik Baik Pemrograman

Sambil kalian mempelajari buku ini, akan ada sisipan praktik baik pemrograman. Praktik baik ini kalian terapkan dalam kegiatan pemrograman seterusnya, termasuk pada topik dan konsep yang belum tercakup dalam aktivitas ini, agar terkumpul dengan rapi. Kalian boleh menambahkan praktik baik lain yang kalian anggap perlu. Tuliskan praktik baik kalian dalam Buku Kerja kalian.

Pemrogram profesional harus menaati aturan-aturan yang lebih ketat. Beberapa perusahaan seperti Google bahkan mempunyai konvensi (kesepakatan) dan juga dikaitkan dengan bahasa pemrograman yang dipakai. Penasaran? Silakan lihat contohnya di google.github.io/styleguide.

Selama kelas X, kalian akan menuliskan banyak program kecil-kecil yang perlu kalian simpan untuk bahan belajar. Jika kalian perlukan, dapat kalian pakai lagi potongan kodenya untuk mempercepat kalian melakukan tugas yang lebih besar. Sebuah program besar biasanya berasal dari potongan-potongan program kecil yang pernah dibuat sebelumnya. Pemrogram tidak perlu mengetik ulang lagi karena potongan kode tersebut sudah pernah diuji. Program kecil ini ibarat potongan *puzzle* yang kelak dapat kalian ambil dan rangkai menjadi sesuatu yang besar.

Berikut ini praktik baik dalam memprogram serta mengelola direktori dan berkas tempat menyimpan kode program:.

1. Selalu menuliskan nama file tempat menyimpan potongan kode sebagai baris pertama kode, dalam bentuk komentar.
2. Menuliskan untuk apa potongan kode program ditulis, akan memudahkan mengambilnya tanpa harus membaca semua kode. Ingat bahwa sama halnya dengan belajar menulis, kalian akan menulis program yang ukurannya (banyaknya baris) makin lama makin besar.
3. Menuliskan komentar yang perlu untuk sekumpulan kode. Pada contoh program di atas, terdapat bagian komentar untuk menuliskan apa yang dilakukan oleh program. Komentar tidak akan dieksekusi oleh kompilator karena fungsinya untuk membantu kalian memahami kode program. Komentar dapat ditulis dalam dua cara.

- a. Ditulis dalam satu baris atau lebih dan diawali dengan `/*` dan diakhiri dengan `*/`.
- b. Ditulis dalam satu baris dengan awalan `//`.

Kode program di atas dapat kalian berikan komentar seperti berikut.

```
/* Program Pertamaku: */
/* Membuat program untuk mencetak "Halo Dunia!" */

#include<stdio.h>
int main(){                // Fungsi utama program
    printf("Halo Dunia!\n"); // Mencetak Halo Dunia! ke layar
    return 0;              // Program berjalan dengan benar
}
```



Jawablah pertanyaan tersebut dalam Lembar Refleksi pada Buku Kerja, dan jangan lupa mencatat kegiatan dalam Jurnal.

1. Apakah kalian sudah pernah membuat program sebelumnya?
2. Saat mengetikkan kode program tersebut, apakah kalian merasa dapat memahami makna dari setiap baris kode tersebut?
3. Apa kesulitan yang kalian rasakan saat mengetikkan program tersebut?
4. Pelajaran paling berkesan apa yang kalian dapatkan dari pertemuan ini?

2. Struktur Program Bahasa C

Bahasa C merupakan bahasa yang terstruktur. Beberapa struktur dasar pada program bahasa C telah terlihat pada program yang kalian tulis sebelumnya. Untuk memudahkan penjelasan, kode program berikut akan diberi nomor baris di bagian kiri.

Penting: Jika kalian ingin mencoba menjalankan contoh program dengan nomor baris seperti di bawah ini, kalian tidak perlu mengetikkan nomor baris tersebut.

```
#include<stdio.h>
int main(){
    printf("Halo Dunia!\n");
    return 0;
}
```

Baris pertama merupakan suatu pernyataan yang digunakan untuk memasukan sebuah *header file* yang bernama `stdio.h`. *Header file* merupakan kumpulan fungsi-fungsi dasar yang dikelompokkan berdasarkan kegunaannya

dan dapat digunakan untuk membantu kalian membuat program. Pada kode di atas, *header file* yang digunakan ialah `stdio.h` yang berisi fungsi-fungsi terkait masukan dan keluaran standar (atau standard input output). Dengan menggunakan `stdio.h`, program yang kalian buat dapat membaca dan menulis data.

Pada baris 2-5, kalian menemukan sebuah blok program yang merupakan implementasi suatu fungsi bernama `main`. Fungsi ini merupakan fungsi yang akan dijalankan pertama kali pada saat program dijalankan. Isi dari blok fungsi tersebut diapit dengan tanda kurung kurawal. Di dalam fungsi ini, pada baris 3, terdapat pemanggilan sebuah fungsi bernama `printf` yang digunakan untuk mencetak suatu data ke layar. Dalam hal ini, data yang ditampilkan ialah sebuah kalimat “Halo Dunia!\n” yang diberikan pada parameter fungsi tersebut. Karakter `\n` yang ada pada kalimat tersebut akan dicetak oleh program sebagai baris baru (*newline*), seperti jika kalian menekan tombol enter pada papan ketik kalian.

Pada baris keempat, terdapat sebuah pernyataan `return 0`. Penjelasan mengenai fungsi pernyataan ini cukup berat untuk dijelaskan pada bagian ini. Singkatnya, pernyataan ini merupakan tanda bagi sistem operasi untuk mengecek selesainya program dengan benar. Apabila program berjalan dan berhenti dengan benar, sistem operasi akan memperoleh nilai 0. Nilai selain 0 akan menjadi tanda bahwa program tidak berhenti dengan benar. Untuk saat ini, yang perlu kalian ketahui ialah pernyataan ini perlu diletakkan di akhir blok fungsi `main` pada program yang kalian tulis.

Kalian dapat memodifikasi kode program tersebut dan menjalankannya kembali dengan melakukan beberapa perubahan, misalnya:

1. Ubah kalimat Halo Dunia menjadi Hello World!
2. Cetak dua baris kalimat di layar, yaitu “Halo Dunia!” dan “Saya siap belajar pemrograman!”

a. Komponen Program dalam Bahasa C

Sebelum membuat program yang lebih kompleks, yang melibatkan ekspresi, struktur kontrol keputusan, atau struktur kontrol perulangan, kalian perlu memahami makna dari berbagai komponen penyusun bahasa C, di antaranya meliputi kata kunci, identifier, variabel, tipe data, dan konstanta.

1) Kata Kunci (*Keyword*)

Kata kunci merupakan kata yang telah memiliki makna khusus yang tidak dapat diubah oleh pemrograman. Dengan kata lain, kalian tidak dapat menggunakan sebagai suatu identifier. Terdapat 32 kata kunci standar pada bahasa C, yaitu:

auto, double, int, struct, break, else, long, switch, case, enum, register, typedef, char, extern, return, union, continue, for, signed, void, do, if, static, while, default, goto, sizeof, volatile, const, float, short, dan unsigned.

Bayangkan, dengan hanya 32 kata kunci, kalian bisa menulis program apa saja dalam bahasa C!

2) Identifier

Identifier adalah nama unik yang dapat kalian ingat yang diberikan ke dalam entitas program C, seperti variabel dan fungsi. Identifier terdiri atas serangkaian karakter dengan aturan berikut.

- Tidak boleh sama dengan kata kunci (*keyword*) dalam bahasa C.
- Disusun dari kombinasi huruf (besar dan kecil), angka, dan underscore “_”.
- Harus dimulai dengan huruf atau *underscore*.
- Bersifat *case-sensitive*, atau sensitif terhadap huruf besar atau kecil (kapitalisasi karakter). Dengan kata lain, sisi dan Sisi akan dianggap sebagai dua identifier yang berbeda.

Praktik Baik Pemrograman

Walaupun dengan aturan di atas kalian dapat membuat *identifier* dengan sangat bebas, keterbacaan kode program menjadi penting. Beberapa praktik baik yang perlu kalian ketahui untuk menghindari kebingungan dalam membaca dan menulis kode ialah seperti berikut.

- Nama yang diawali oleh underscore digunakan untuk keperluan tertentu dan tidak seharusnya digunakan dalam membuat program di unit ini.
- Nama variabel dan fungsi harus ditulis dengan huruf non-kapital. Akan tetapi, apabila terdiri atas dua atau lebih kata, kalian dapat menggunakan teknik menulis dengan standar camel case. Huruf awal kata, selain kata pertama, ditulis menggunakan huruf kapital. Contoh: `totalHargaBarang`, `namaMahasiswa`.
- Hindari menggunakan *identifier* yang sangat mirip dalam satu kode program. Misalnya, `totalHarga` dengan `total_harga`. Hindari juga dua atau lebih variabel yang hanya berbeda di kapitalisasi seperti `totalharga` dan `totalHarga`.
- Hindari *identifier* yang terlihat mirip, misalnya karakter ‘I’, ‘1’, dan ‘l’ terlihat sangat mirip satu sama lain. Catatan: karakter yang disebutkan ialah huruf i kapital, angka 1, dan huruf L non-kapital.
- Identifier haruslah bersifat *mnemonic*. Untuk variabel, artinya identifier variabel tersebut harus menunjukkan isi dari variabel tersebut. Untuk fungsi, *identifier* menunjukkan apa yang dilakukan oleh fungsi tersebut.

6. Dalam dunia pemrograman, ada beberapa nama variabel yang menjadi kebiasaan untuk dipakai, misalnya nama variabel untuk mengunjungi elemen tabel, dipakai indeks *i*, *j*, dan *k*.
7. Konstanta dinamai dengan huruf kapital. Misalnya, *PI*.

3) Tipe Data

Komputer dapat mengolah data yang beragam. Pada dasarnya, data yang diolah oleh komputer, baik berupa numerik ataupun karakter, akan disimpan dalam bentuk biner. Oleh karena itu, nilai yang kalian masukkan dalam komputer pastilah akan disimpan dalam bentuk biner. Program perlu mengetahui bagaimana bilangan biner dibaca sehingga diperlukan suatu mekanisme untuk memberi tahu program tentang data yang kita simpan pada variabel tersebut. Hal ini diakomodir melalui tipe data. Suatu tipe data akan memiliki nama tipe, jenis data yang disimpan, dan rentang yang berbeda. Pada bahasa C, terdapat beberapa tipe data dasar yang dapat digunakan (Tabel 7.3).

Tabel 7.3 Beberapa Tipe Data Pada Bahasa C Beserta Ukuran Memori dan Rentang Nilainya

Nama Tipe	Jenis Data	Ukuran Memori	Rentang
int	Bilangan bulat	4 byte	-2.1×10^9 hingga 2.1×10^9
short	Bilangan bulat	2 byte	-32768 hingga 32767
long	Bilangan bulat	8 byte	-9.2×10^8 hingga 9.2×10^8
float	Bilangan riil	4 byte	1.2×10^{38} hingga 3.4×10^{38}
double	Bilangan riil	8 byte	2.3×10^{308} hingga 1.7×10^{308}
char	Karakter*	1 byte	-127 hingga 128

* Seperti yang termuat pada kode ASCII: ascii-code.com

Perhatikan bahwa rentang yang diberikan memungkinkan nilai negatif hingga positif, atau disebut tipe *data signed*. Apabila kalian menambahkan kata kunci *unsigned* di depan tipe data, tipe data tersebut hanya akan menampung bilangan positif dengan rentang dari 0 hingga $2^{\text{jumlah bit}} - 1$.

Praktik Baik Pemrograman

Gunakan tipe data yang sesuai dengan kebutuhan kalian. Sebagai contoh, saat mengolah data usia manusia dalam satuan tahun, kalian cukup menggunakan tipe *data short* yang memerlukan memori lebih kecil. Ketika kalian nanti membuat program yang mengolah dan menyimpan data dengan jumlah yang sangat besar, praktik ini dapat membuat program kalian berjalan dengan kebutuhan memori yang lebih efisien.

4) Variabel

Pada matematika, kalian mengenal variabel sebagai sebuah wadah untuk menyimpan suatu nilai. Variabel pada program memiliki fungsi yang sama. Nilai yang diberikan pada sebuah variabel akan disimpan di memori komputer. Komputer memberikan alamat pada lokasi memori tersebut yang sulit diingat oleh manusia. Oleh karena itu, variabel diberikan nama simbolik yang mudah untuk diingat oleh kalian dengan menggunakan *identifier*.

Dalam bahasa C, variabel perlu dideklarasikan dengan memberikan tipe data dan *identifiers* sebelum dapat digunakan. Deklarasi dapat dilakukan dengan menggunakan pernyataan berikut:

```
<tipe_data><nama_variabel>;
```

Pada saat deklarasi, variabel juga dapat diberikan nilai awal, misalnya dalam bentuk:

```
<tipe_data><nama_variabel> = <nilai_awal>;
```

Variabel dengan tipe yang sama dapat dideklarasikan secara ringkas seperti:

```
<tipe_data><nama_variabel1>, <nama_variabel2>;
```

Beberapa contoh untuk mendeklarasikan variabel dapat dilihat pada Tabel 7.4.

Tabel 7.4 Beberapa Contoh Deklarasi Variabel

Tipe Data	Identifier	Deklarasi	Deklarasi dengan Nilai Awal
<i>int</i>	totalHarga	int totalHarga;	int totalHarga = 150000;
<i>short</i>	usia	short usia;	short usia = 29;
<i>long</i>	jumlahAtom	long jumlahAtom;	long jumlahAtom = 9123151252214;
<i>float</i>	jarak	float jarak;	float jarak = 2.28;
<i>double</i>	galat	double galat;	double galat = 0.0000000001234;
<i>char</i>	huruf	char huruf;	char huruf = `a`;

Deklarasi secara ringkas misalnya dapat dilakukan seperti berikut:

```
int panjang = 1, lebar = 2, luas;  
float alas, sisi, volume;
```

Tempat deklarasi variabel akan berpengaruh pada penggunaan variabel tersebut. Apabila deklarasi variabel dilakukan di dalam sebuah fungsi, variabel tersebut hanya dapat digunakan di dalam fungsi tersebut. Variabel seperti ini disebut variabel lokal. Apabila deklarasi dilakukan di luar fungsi, variabel tersebut akan dapat diakses di bagian program mana pun. Variabel ini disebut variabel global.

5) Konstanta

Berbeda dengan variabel yang nilainya dapat berubah, konstanta tidak dapat diubah. Saat dideklarasikan, nilai dari konstanta diberikan dan tidak dapat diubah kembali. Apabila kalian memaksa mengubah konstanta, kompilator akan memberikan pesan kepada kalian. Penggunaan konstanta yang lazim ialah untuk menyimpan nilai konstan seperti *pi* (π), *rho* (ρ), dan konstanta lainnya yang lazim digunakan.

Konstanta dapat dideklarasikan seperti variabel, dengan menambah kata kunci `const` di depan tipe data. Nilai awal harus langsung diberikan pada saat deklarasi. Misalnya, deklarasi konstanta *pi* dapat dilakukan sebagai berikut:

```
const float PI = 3.14;
```

6) Membaca dan Menulis

Untuk dapat membantu manusia, program harus dilengkapi dengan kemampuan berkomunikasi. Ada banyak cara untuk berkomunikasi lewat antarmuka pengguna (*user interface*), tetapi bentuk komunikasi dasar yang perlu kalian kuasai komunikasi lewat *command line interface* (CLI). Lewat CLI, kalian dapat berkomunikasi dengan sebuah program menggunakan teks, dan program pun akan merespons kalian dengan menggunakan teks. Dengan kata lain, interaksi menggunakan CLI sangat bergantung pada kemampuan program untuk membaca data yang diberikan oleh pengguna dan menuliskan hasil pekerjaan.

Agar dapat membaca dan menulis, program yang kalian buat perlu menggunakan *header* **stdio.h** yang memuat fungsi masukan-keluaran standar menggunakan CLI. Dua fungsi utama yang dapat digunakan ialah *scanf* untuk membaca dan *printf* untuk menulis. Untuk lebih jelasnya, perhatikanlah contoh program berikut. Pada program baca tulis berikut, kalian akan memerintahkan komputer untuk membaca suatu bilangan dan menuliskannya kembali.

```
/* Program Baca Tulis 3 */

#include<stdio.h>
int main(){
    int bilangan;
    scanf("%d", &bilangan);
    printf("Bilangan yang dibaca bernilai: ");
    printf("%d.\n", bilangan);
    return 0;
}
```

Pada baris keempat, program memanggil fungsi *scanf* untuk membaca masukan dari pengguna. Pada saat baris ini dieksekusi, program akan berhenti hingga pengguna memasukkan suatu bilangan dan menekan tombol *enter*.

Perhatikan bahwa pada baris tersebut, fungsi `scanf` menerima dua buah parameter, yaitu `%d` yang merupakan spesifikasi format (*format specifier*) dan `&bilangan` yang merupakan variabel untuk menampung nilai yang dibaca. Artinya, pada saat kalian menekan *enter*, program akan membaca nilai 10 yang kalian masukkan sebagai sebuah nilai bertipe data `int`, dan akan menyimpannya ke variabel `bilangan`. Di depan `bilangan`, *terdapat tanda ampersand (& yang wajib digunakan untuk melakukan pembacaan*. Makna dari simbol `&` akan dijelaskan lebih detail pada kesempatan lain.

Setiap tipe data memiliki spesifikasi format yang dapat digunakan untuk menjelaskan jenis data kepada program. Ingat, program membaca dan menyimpan semua data sebagai bilangan biner. Spesifikasi format yang digunakan untuk tipe data pada bahasa C dapat dilihat pada Tabel 7.5.

Tabel 7.5 Spesifikasi Format pada Bahasa C

Nama Tipe	Spesifikasi Format (Signed)	Spesifikasi Format (Unsigned)
<code>int</code>	<code>%d</code>	<code>%ud</code>
<code>short</code>	<code>%d</code>	<code>%ud</code>
<code>long</code>	<code>%ld</code>	<code>%uld</code>
<code>float</code>	<code>%d</code>	<code>%ud</code>
<code>double</code>	<code>%ld</code>	<code>%uld</code>
<code>char</code>	<code>%c</code>	<code>%uc</code>

Baris kelima dan keenam adalah pernyataan untuk menulis menggunakan fungsi `printf`. Pada baris kelima, kalian memerintahkan program untuk mencetak suatu kalimat yang diapit dengan tanda petik ganda. Pada baris keenam, kalian memerintahkan program untuk mencetak nilai dari variabel `bilangan`. Mirip seperti fungsi `scanf`, kalian menemukan adanya spesifikasi format dan variabel. Bedanya, di depan variabel, tidak perlu ada tanda ampersand.

Saat dipanggil, program akan mencetak nilai dari variabel `bilangan` dengan format yang diberikan. Hanya saja, ada karakter baru yang muncul, yaitu `\n`. Ini adalah *escape sequence* yang digunakan untuk membuat garis baru, persis seperti ketika kalian menekan tombol *enter* pada aplikasi pengolah kata. Data yang ditulis setelah `\n` akan dicetak di baris yang baru oleh program.

Ada beberapa *escape sequence* lain yang dapat digunakan, yaitu:

```
\n (newline)
\t (tab)
\v (vertical tab)
```

```

\f (new page)
\b (backspace)
\r (carriage return)
\n (newline)
\a (beep, bell)
\\ (garis miring)
\" (tanda kutip ganda)
\` (tanda kutip)

```

Tentunya, kalian juga menggunakan fungsi `printf` untuk mencetak tipe data lain, misalnya bilangan riil. Pada bilangan riil, kalian dapat membatasi jumlah digit di belakang desimal dengan memodifikasi spesifikasi format. Perhatikan kode berikut yang akan menampilkan angka 12.345 ke layar.

```

/* Program Cetak Desimal */
#include <stdio.h>

int main() {
    float x=12.3456789;
    printf("%.3f\n", x);
    return 0;
}

```

Kalian juga dapat membaca atau menulis dua atau lebih nilai sekaligus seperti pada contoh berikut:

```

/* Program Cetak Bilangan Bulat dan Desimal */
#include <stdio.h>

int main() {
    int a, b;
    float c;

    scanf("%d %d %f", &a, &b, &c);
    printf("%d %d %.3f\n", a, b, c);
    return 0;
}

```



Ketikkan kode tersebut, berikan masukan 10 10 10.1234, dan perhatikan hasilnya.

Praktik Baik Pemrograman

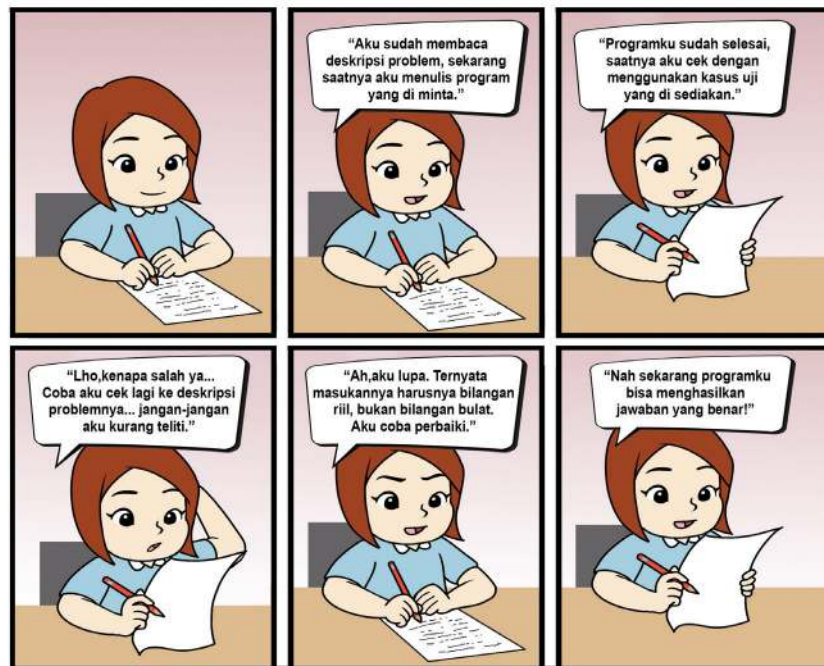
Seiring dengan makin kompleksnya program yang kalian buat, ada beberapa praktik baik yang dapat kalian lakukan.

1. Belajar menulis kode program memerlukan kemampuan bereksperimen dan mencoba hal baru. Karena keterbatasan halaman, buku ini tidak mungkin memberikan contoh selengkap mungkin untuk mencoba berbagai kemungkinan dalam menulis kode program. Oleh karena itu, kalian perlu melakukan eksperimen dan mencoba memodifikasi kode program yang telah diberikan. Kalian dapat membaca dokumentasi bahasa C yang dapat diakses di devdocs.io/c ketika menemukan suatu konsep ingin kalian pelajari lebih lanjut.
2. Jangan takut apabila program kalian gagal berjalan karena hal itu sangat wajar ketika belajar menulis kode program. Bahkan, pemrogram senior pun masih bisa melakukan kesalahan. Jadikan hal tersebut sebagai pemacu untuk mencari tahu hal yang membuat program kalian gagal berjalan. Dari situ, kalian akan menambah pengalaman dan di masa depan tidak akan kembali melakukan kesalahan yang sama.
3. Buat kode program kalian lebih mudah dibaca dengan menggunakan *whitespace* atau ruang kosong. *Whitespace* adalah istilah bagi karakter yang tidak tampak di layar (contoh: spasi, tab, dan newline). *Whitespace* dapat digunakan untuk membuat kode kalian lebih mudah dibaca dengan memberikan jarak pada kode. Jarak biasanya diberikan antara kumpulan baris kode yang memiliki peran berbeda. Contoh-contoh kode program yang diberikan pada unit ini disajikan dengan menggunakan *whitespace* yang dapat kalian adopsi.

Semangat terus untuk bereksperimen!

3. Belajar Menulis Program Sambil Menyelesaikan Masalah

Pada unit ini, aktivitas Ayo, Kita Berlatih diberikan sebagai bentuk latihan menyelesaikan suatu problem yang diberikan dengan pemrograman. Karena unit ini bersifat pengenalan pada kegiatan menulis kode program (coding), problem yang diberikan pada kalian diberikan dalam bentuk spesifikasi yang telah terstruktur. Pada kenyataannya, ketika kalian membuat program untuk menyelesaikan suatu permasalahan nyata, kalian perlu membuat sendiri spesifikasi dari program yang akan dibuat. Kalian perlu menetapkan sendiri tujuan dari program, format masukan, serta format keluaran dari program. Aktivitas ini akan kalian lakukan di akhir unit, tetapi sekarang kalian dapat fokus membuat program berdasarkan spesifikasi yang telah diberikan. Selain itu, pada unit pratik lintas bidang (PLB), kalian akan diajak untuk membuat sebuah program untuk menyelesaikan permasalahan di lingkungan tempat kalian tinggal.



(Sumber: Dokumen Kemendikbud, 2021)

Pada setiap aktivitas, kalian akan diberikan setidaknya satu permasalahan untuk diselesaikan dengan menggunakan konsep yang telah diberikan pada buku. Bentuk permasalahan tersebut akan diberikan dengan struktur pada Tabel 7.6.

Tabel 7.6 Spesifikasi Program yang Diberikan pada Problem Latihan

Bagian	Penjelasan
Nomor dan Nama Problem	Identitas dari permasalahan.
Deskripsi Soal	Memberikan konteks permasalahan yang perlu diketahui oleh siswa dalam membuat program.
Format Masukan	Memberikan susunan data yang diberikan pada program oleh pengguna, beserta ukuran dari data tersebut.
Format Keluaran	Memberikan susunan informasi yang akan dikeluarkan oleh program kepada pengguna.
Contoh Masukan	Memberikan contoh data yang dimasukkan.
Contoh Keluaran	Memberikan contoh informasi yang dikeluarkan program berdasarkan data yang dimasukkan.

Bagian format keluaran dan format masukan menjadi penting karena auto-grader (jika kalian menggunakan ini di kelas) akan memeriksa kode program kalian dengan sangat ketat berdasarkan pasangan masukan-keluaran yang telah disiapkan, atau disebut kasus uji (test case). Apabila program kalian membaca

atau menulis dengan format yang salah, auto-grader tidak akan menganggap program kalian benar. Untuk membantu kalian memahami format tersebut, satu atau lebih contoh masukan dan keluaran akan diberikan.

Praktik Baik Pemrograman

Problem akan memberikan beberapa kasus uji. Akan tetapi, bukan berarti kalian hanya perlu mengecek program kalian dengan kasus uji yang ada di buku. Kalian perlu menguji program kalian dengan menggunakan berbagai kasus uji, yang kalian buat sendiri. Beberapa problem dirancang dengan adanya jebakan yang dapat membuat program kalian gagal berjalan pada kasus tertentu. Misalnya, ketika masukan yang diberikan memiliki rentang yang besar atau adanya kondisi yang dapat menyebabkan program berhenti.

Ayo, Kita Berlatih 3: Menulis dan Memperbaiki Program

Problem 1: Belajar Baca Tulis (Tingkat Kesulitan: ★ ★)

Deskripsi Soal:

Saatnya kalian berlatih untuk menulis kode program. Kalian akan membuat program untuk membaca tiga jenis bilangan dan mencetaknya kembali ke layar dengan format yang diberikan.

Format Masukan:

Satu baris yang berisi tiga buah bilangan yang dipisahkan oleh spasi. Bilangan pertama merupakan bilangan bulat positif dengan nilai maksimum 10,000. Bilangan kedua merupakan bilangan riil positif dengan nilai maksimum 10.0000. Bilangan ketiga merupakan bilangan bulat positif dengan nilai maksimum 100.

Format Keluaran:

Ada tiga baris. Baris pertama berisi bilangan pertama. Baris kedua berisi bilangan kedua dengan dua bilangan di belakang desimal. Baris ketiga berisi bilangan ketiga.

Contoh Kasus Uji

Masukan	Keluaran
10 20.1235 30	10 20.12 30

Problem 2: Bantulah Intan! (Tingkat Kesulitan: ★ ★ ★)

Intan ingin membuat program untuk mencetak kalimat berikut ke layar.

Andi berkata, "Satu, dua, tiga!".

Lalu, Andi pun menendang bola tersebut.

Akan tetapi, Intan tidak berhasil mencetak kalimat tersebut ke layar dengan persis sama. Berikut ini program yang ditulis oleh Intan.

```
#include <stdio.h>
int main(){
    printf("Andi berkata, \"Satu, dua, tiga!\".");
    printf("Lalu, Andi pun menendang bola tersebut.\n");
    return 0;
}
```

Perbaikilah program tersebut hingga berhasil mencetak kalimat yang benar!

Problem 3: Salah Baca (Tingkat Kesulitan: ★ ★)

Temukanlah kesalahan pada kode program berikut, kemudian perbaikilah kode berikut hingga dapat menghasilkan jawaban yang benar.

```
#include <stdio.h>

int main(){
    scanf("%c %c", a, b);
    printf("Bilangan pertama: %c\\d", a);
    printf("Bilangan kedua: %c\\d", b);
    return 0;
}
```



Ayo Kita Renungkan

Jawablah pertanyaan berikut dalam Lembar Refleksi pada Buku Kerja. Jangan lupa mencatat kegiatan dalam Jurnal.

1. Pada bagian ini, kalian mendapatkan banyak konsep baru tentang program. Seperti apa perasaan kalian saat ini?
2. Apakah kalian bereksperimen dengan contoh-contoh yang diberikan di buku? Jika ya, pengetahuan paling menarik apa yang kalian temukan dari hasil eksperimen tersebut?
3. Pada saat memperbaiki program, apakah kalian dapat menemukan kesalahan pada program dengan mudah? Apakah kalian sebelumnya pernah melakukan kesalahan tersebut pada saat menulis program?
4. Bagaimana rencana kalian untuk menulis program setelah pertemuan ini? Apakah kalian sudah menentukan rencana berlatih secara mandiri agar dapat menulis program dengan lebih lancar dan terampil?

4. Ekspresi

Ekspresi merupakan bagian yang tidak terpisahkan dari program. Di dalam matematika, ekspresi terdiri atas kombinasi beberapa *operand* dan operator yang memiliki makna. Kalian telah terbiasa menulis ekspresi pada matematika,

misalnya dalam penjumlahan $10 + 5$ yang melibatkan dua buah operand (10 dan 5) dan sebuah operator (+). Ekspresi pada pemrograman mirip dengan ekspresi yang kalian pelajari pada matematika, tetapi diperkaya dengan tambahan operator-operator untuk memudahkan kalian dalam menulis program.

Berdasarkan jumlah operand, suatu ekspresi dapat dibagi menjadi ekspresi:

1. *Unary* (satu *operand*), misalnya $-a$ untuk menegasikan suatu bilangan.
2. *Binary* (dua *operand*), misalnya $a+b$ untuk penjumlahan.
3. *Ternary* (tiga *operand*), misalnya $a ? b : c$ yang akan dijelaskan lebih rinci.

Berdasarkan fungsinya, operator dalam bahasa C dapat dibagi menjadi operator penugasan, operator aritmatika, operator logika, serta *increment* dan *decrement*.

a. Operator Penugasan

Operator penugasan (simbol '=') untuk memberikan suatu nilai konstanta atau nilai yang diperoleh dari suatu ekspresi ke dalam variabel. *Operand* di sisi kiri akan menerima nilai dari *operand* di sisi kanan operator penugasan. Contoh penugasan telah kalian lakukan pada saat melakukan deklarasi variabel seperti:

```
int a = 5;
```

Perhatikan dan ingat baik-baik bahwa makna dari simbol '=' dalam bahasa C sangat berbeda dengan tanda “sama dengan” dalam matematika!

b. Operator Aritmatika

Operator aritmatika digunakan untuk melakukan operasi matematika yang kalian kenal. Ada perbedaan notasi penulisan operator dengan yang kalian pelajari pada mata pelajaran Matematika yang dapat dilihat pada Tabel 7.7.

Tabel 7.7 Operator Matematika pada Bahasa C

Operasi Matematika	Contoh Ekspresi Aljabar	Operator Bahasa C	Ekspresi Bahasa C
Penjumlahan	$1 + 2$	+	$1 + 2$
Pengurangan	$a - b$	-	$1 - b$
Perkalian	ab	*	$a * b$
Pembagian	a/b	/	a / b
Modulo	$a \bmod b$	%	$a \% b$

Penting: Hati-hati dalam melakukan pembagian pada bahasa C. Bahasa C sangat sensitif terhadap tipe data sehingga pembagian dua buah bilangan bulat akan menghasilkan bilangan bulat. Sebagai contoh, $10/3$ pada kode program C saat dieksekusi akan menghasilkan nilai 3.

Seperti pada matematika, suatu ekspresi dapat terdiri atas banyak operator. Agar ekspresi tersebut dieksekusi dengan benar, diperlukan suatu urutan operasi (atau *operator precedence*). Urutan operasi dari operator pada bahasa C ialah:

1. Tanda kurung ()
2. Operator perkalian ‘*’, pembagian ‘/’, dan modulo ‘%’.

Selain operator-operator matematika tersebut, C juga memiliki operator-operator lain yang akan dipelajari di lain kesempatan. Daftar lengkap semua operator pada C dapat dilihat di http://en.cppreference.com/w/c/language/operator_precedence

Hasil ekspresi dapat disimpan dalam suatu variabel menggunakan operator penugasan. Misalnya:

```
a = 5;  
b = a + 1;  
b = b + 5;  
c = (a + b) * 2 + a;  
d = c % b;
```

Ekspresi seperti $b = b + 5$ mungkin tidak pernah kalian temui pada mata pelajaran Matematika. Akan tetapi, ekspresi ini sangat lazim ditemukan dalam menulis kode program. Makna dari ekspresi ini ialah kalian melakukan perubahan pada nilai suatu variabel berdasarkan nilai variabel pada saat ini. Hal ini telah kalian lakukan pada saat melakukan penelusuran pada Diagram Alir 4 di bagian awal unit ini.

Untuk ekspresi seperti ini, bahasa C menyediakan bentuk yang lebih ringkas yang disebut *compound assignment operator* atau operator penugasan majemuk. Misalnya, $b = b + 5$ dapat disingkat menjadi $b += 5$. Hal ini juga berlaku untuk operator matematika lainnya seperti $b *= 2$, $c /= 3$, dan seterusnya.

c. Operator Increment dan Decrement

Selain operator penugasan majemuk pada bagian sebelumnya, bahasa C juga memberikan operator *unary* yang lebih khusus untuk menambah (++) atau mengurangi nilai suatu variabel (--) dengan angka 1. Operator ini dapat diletakkan sebelum (*prefix*) atau setelah (*postfix*) operand.

Pada penulisan dalam bentuk prefix, perubahan nilai akan langsung dilakukan pada nilai variabel sebelum nilai variabel tersebut digunakan pada ekspresi. Sebagai contoh, setelah dua baris kode berikut dijalankan, nilai x akan bernilai 2 dan y akan bernilai 1.

```
int x = 1;
int y = ++x;
```

Sebaliknya, pada penulisan dalam bentuk *postfix*, nilai variabel akan digunakan terlebih dahulu pada ekspresi, baru perubahan dilakukan. Sebagai contoh, setelah dua baris kode berikut dijalankan, nilai x dan y akan bernilai 1.

```
int x = 1;
int y = x++;
```

d. Operator Logika, Relasional, dan Kesamaan

Di samping operator aritmatika, juga dikenal operator logika, relasional, dan kesamaan. Ekspresi yang menggunakan operator ini akan memiliki nilai benar (true atau dalam bahasa C bernilai tidak sama dengan 0) atau salah (false atau bernilai 0). Operator pada kategori ini memiliki peran yang sama dengan operator logika, relasional, dan kesamaan pada mata pelajaran Matematika. Operator tersebut pada bahasa C dapat dilihat pada Tabel 7.8.

Tabel 7.8 Operator Logika, Relasional, dan Kesamaan Pada Bahasa C

Aljabar	Bahasa C	Contoh	Makna
Operator Kesamaan / Pertidaksamaan			
=	==	a == b	Apakah nilai a sama dengan b?
≠	!=	a != b	Apakah nilai a tidak sama dengan b?
Operator Relasional			
>	>	a > b	Apakah a lebih besar dari b?
<	<	a < b	Apakah a lebih kecil dari b?
≥	>=	a >= b	Apakah a lebih besar atau sama dengan b?
≤	<=	a <= b	Apakah a lebih kecil atau sama dengan b?
Operator Logika			
AND	&&	a > 0 && b > 0	Apakah a dan b lebih besar daripada 0?
OR		a > 0 b > 0	Apakah a atau b lebih besar daripada 0?
NOT	!	!(a > 0)	Apakah a tidak lebih besar daripada 0?

e. Operator Kondisional (*Ternary*)

Operator kondisional adalah operator ternary yang akan mengembalikan nilai berdasarkan suatu kondisi tertentu. Misalnya, kalian ingin mengembalikan nilai 1 jika suatu ekspresi $a < 3$ bernilai benar, dan mengembalikan nilai 0 jika sebaliknya. Kalian dapat menuliskannya dalam bentuk

$$n = ((a < 3) ? 1 : 0)$$

Bentuk umum penggunaan operator ini adalah sebagai berikut:

(kondisi ? ekspresi-1 : ekspresi-2)

Ekspresi 1 akan dikembalikan jika kondisi bernilai benar, sedangkan ekspresi 2 dikembalikan jika kondisi bernilai salah.

Praktik Baik Pemrograman

Pada saat menuliskan suatu ekspresi, gunakanlah pengelompokan dengan tanda kurung untuk membuat ekspresi tidak ambigu dan mudah untuk dibaca. Pastikan ekspresi tersebut ditulis dengan benar sebelum kalian menjalankan program. Biasakan untuk menguji (dalam pikiran kalian) sambil menulis, tidak menunggu program dijalankan.

Ayo, Kita Berlatih 4: Latihan Ekspresi

Problem 1: Menghitung Luas Tanah (Tingkat Kesulitan: ★ ★)

Deskripsi Soal:

Pak Algor memiliki sebidang tanah berbentuk segitiga siku-siku. Ia ingin mengetahui berapa luas tanah yang ia miliki. Bantulah Pak Algor menghitung dengan membuat program untuk menghitung luas tanahnya.

Format Masukan:

Dua buah bilangan bulat yang merupakan alas dan tinggi segitiga dari tanah Pak Algor.

Format Keluaran:

Sebuah bilangan riil, dengan dua digit di belakang desimal yang merupakan luas tanah Pak Algor. Akhiri keluaran dengan karakter newline.

Contoh Kasus Uji

Masukan	Keluaran
20 40	400.00

Problem 2: Menghitung Luas persegi (Tingkat Kesulitan: ★ ★)

Simak kembali Diagram Alir 1: Menghitung luas persegi pada bagian algoritma di unit ini. Buatlah program berdasarkan diagram alir tersebut. Panjang sisi yang diberikan ialah sebuah bilangan riil dengan nilai maksimum 1000 cm.

Problem 3: Hasil Bagi dan Sisa Pembagian (Tingkat Kesulitan: ★ ★ ★)

Deskripsi Soal:

Buatlah sebuah program untuk menampilkan hasil dan sisa pembagian dari dua buah bilangan bulat positif a dan b.

Format Masukan:

Dua buah bilangan positif a dan b yang dipisahkan oleh karakter spasi. Keduanya bernilai paling besar 10 miliar.

Format Keluaran:

Dua buah bilangan bulat yang ditulis di baris berbeda. Baris pertama adalah hasil pembagian, sedangkan baris kedua adalah sisa pembagian. Hasil pembagian dibulatkan ke bawah.

Contoh Kasus Uji

Masukan	Keluaran
1000 3	333 1

Problem 4: Benar atau Salah? (Tingkat Kesulitan: ★ ★ ★)

Ekspresi yang memuat operator logika, relasional, dan kesamaan dapat kalian telusuri tanpa menjalankan program. Berikut ini, diberikan beberapa ekspresi yang perlu kalian cek nilainya, jika diketahui nilai a = 1, b = 2, dan c = 3.

No	Ekspresi	Hasil
1	(a < b) (b > c)	
2	(a >= b) (b != c)	
3	(b == a) && (c > a)	
4	(b >= a) && ((b < c) (c > a))	

Problem 5: Percantik Kode Program Ini! (Tingkat Kesulitan: ★ ★ ★ ★)

Uh... Kalian baru saja mendapatkan tugas untuk mempelajari sebuah kode program. Akan tetapi, kode program yang kalian terima tidak ditulis dengan menerapkan praktik baik pemrograman yang telah kalian pelajari sehingga sangat sulit untuk dipahami! Apalagi ternyata, saat kalian jalankan, program ini tidak bisa berjalan dengan benar.

```
#include <studio.h> int main(){ float jr2; float l; float  
O; scanf("%f", jr2); l = 3.14*jr2*jr2; O = 2*3.14*jr2;  
printf("%.2f %.2f\n",l, O); return 0; }
```

Sebagai seorang siswa teladan yang telah mempelajari praktik baik pemrograman, perbaikilah program di atas ke dalam bentuk yang menerapkan praktik baik pemrograman. Kemudian, jika ada, perbaikilah program tersebut sehingga dapat berjalan dengan benar. Setelah itu, apakah kalian dapat menebak apa yang dilakukan oleh program tersebut?



Jawablah pertanyaan berikut dalam Lembar Refleksi pada Buku Kerja. Jangan lupa mencatat kegiatan dalam Jurnal.

1. Pada bagian ini, kalian mendapatkan banyak konsep baru tentang program. Seperti apa perasaan kalian saat ini?
2. Apakah kalian bereksperimen dengan contoh-contoh yang diberikan di buku? Jika ya, pengetahuan paling menarik apa yang kalian temukan dari hasil eksperimen tersebut?
3. Pada saat mengerjakan Problem 3, apakah kalian menyadari bahwa tipe data int tidak dapat menampung nilai hingga 10 miliar? Tipe data apa yang seharusnya digunakan?
4. Apa yang kalian rasakan saat memperbaiki program yang diberikan pada Problem 5? Apakah kalian makin menyadari pentingnya menerapkan praktik baik pemrograman?

5 Struktur Kontrol Keputusan

Pernahkah kalian ingin pergi ke sebuah tempat tertentu dengan menggunakan moda transportasi? Keputusan menggunakan sebuah moda transportasi untuk bepergian biasanya tergantung pada sebuah keadaan tertentu. Misalnya, apabila kondisi hujan, maka kalian akan lebih memilih menggunakan mobil daripada menggunakan sepeda motor, namun apabila cuaca sedang cerah dan jarak yang ditempuh adalah dekat, maka kalian akan memilih menggunakan sepeda motor.

Komputer merupakan alat yang membantu banyak aktivitas manusia. Pada dasarnya, komputer menjalankan perintah dari manusia. Perintah-perintah tersebut dituangkan secara tertulis dalam sebuah aturan tertentu yang disebut sebagai kode program yang bertujuan untuk mengatur bagaimana komputer harus bertindak untuk menyelesaikan sebuah permasalahan tertentu. Hal ini termasuk juga dalam proses pengambilan keputusan, seperti halnya dalam contoh pemilihan moda transportasi di atas.

Pada bagian ini kita akan mempelajari bagaimana pengambilan keputusan dilakukan dalam sebuah program. Istilah yang sering digunakan untuk ini adalah kondisional. Apa itu *kondisional*? Secara sederhana, kondisional adalah sebuah bentuk pernyataan “jika ..., maka ...”. Pernyataan ini dibuat untuk mengekspresikan sebuah aksi berdasarkan sebuah kondisi tertentu. Sebagai contoh, ketika kita diminta untuk mengklasifikasikan sebuah bilangan merupakan bilangan ganjil atau genap, maka dapat kita membuat sebuah aturan sebagai berikut:

1. Jika bilangan tersebut habis dibagi 2, maka bilangan tersebut termasuk bilangan genap.
2. Jika bilangan tersebut tidak habis dibagi 2, maka bilangan tersebut termasuk bilangan ganjil.

Proses tersebut merupakan salah satu ilustrasi dari sebuah pernyataan kondisional. Pada bahasa pemrograman C, ada beberapa jenis pernyataan kondisional, misalnya pernyataan *if-else*, pernyataan *switch-case*, dan pernyataan yang bersarang.

a. Struktur Kontrol Keputusan *If - Else*

Ada beberapa variasi penggunaan struktur kontrol keputusan *If - Else*. Bentuk umum dari pernyataan *if* adalah sebagai berikut.

```
if (kondisi) {  
    <pernyataan>;  
    <pernyataan>;  
    ....  
}
```

Bagian kondisi dapat diisi dengan ekspresi yang menghasilkan nilai benar atau salah. Apabila kondisi menghasilkan nilai benar, semua pernyataan yang berada di dalam struktur kontrol keputusan tersebut akan dieksekusi oleh program. Sekarang, perhatikan program berikut, dan lakukanlah penelusuran untuk memeriksa keluaran dari program tersebut.

```
/* Program Membandingkan Bilangan */  
#include <stdio.h>  
  
int main(){  
    int a = 1, b = 1;  
    if (a == b) {  
        printf("a sama dengan b \n");  
    }  
    return 0;  
}
```

Program tersebut menggunakan struktur keputusan pada baris 6-8. Ekspresi yang digunakan pada bagian kondisi ialah `a==b`, sedangkan pernyataan yang dieksekusi jika kondisi benar terdapat pada baris 7.

```
/* Program Membandingkan Bilangan */  
#include <stdio.h>  
  
int main(){  
    int a = 1, b = 1;  
    if (a == b)  
        printf("a sama dengan b \n");  
    return 0;  
}
```

Struktur kontrol keputusan dapat menambahkan blok *else* yang akan dieksekusi apabila kondisi bernilai salah.

```

    if (kondisi)
<pernyataan>;
    else
<pernyataan>;

```

Misalnya, untuk mengecek apakah suatu bilangan merupakan bilangan ganjil atau genap, kalian dapat memanfaatkan struktur *if-else* sebagai berikut.

```

/* Program Cek Ganjil-Genap */
#include <stdio.h>
int main(){
    int bilangan;
    scanf("%d", &bilangan);
    if (bilangan % 2 == 0)
        printf("Bilangan Genap\n");
    else
        printf("Bilangan Ganjil\n");
    return 0;
}

```

Apabila kondisi makin kompleks, struktur *if-else* ini dapat dikembangkan kembali menjadi:

```

if (kondisi ke-1)
    <statement>
else if (kondisi ke-2)
    <statement>
.....
else if (kondisi ke-n )
    <statement>
else
    <statement>

```

b. Struktur Kontrol Keputusan Switch-Case

Struktur kontrol keputusan yang memiliki cabang banyak dapat dibuat lebih sederhana menggunakan struktur *switch-case*. Bentuk umum dari struktur ini ialah sebagai berikut.

```

switch(switch_expr)
{
    case (constant expr1) :<statement>;
        <statement>;
        break;
    case (constant expr2) :<statement>;
        <statement>;
        break;
    .....
    default                :<statement>;
        <statement>;
        break;
}

```

Sebagai contoh, perhatikan kode program berikut:

```
#include <stdio.h>
int main() {
    int bilangan, sisaPembagian;
    scanf("%d", &bilangan);
    sisaPembagian = bilangan % 4;
    switch (sisaPembagian) {
        case 0: printf("Habis Dibagi\n"); break;
        case 1: printf("Sisa Satu\n"); break;
        case 2: printf("Sisa Dua\n"); break;
        case 3: printf("Sisa Tiga\n"); break;
    }
    return 0;
}
```

Pada program di atas, struktur *switch-case* memeriksa nilai yang ada pada variabel sisa pembagian. Karena nilai tersebut merupakan sisa pembagian sebuah bilangan dengan empat, hanya ada empat kemungkinan nilai, yaitu 0 sampai 3. Setiap kemungkinan nilai tersebut diperiksa melalui empat buah struktur case yang akan mencetak kalimat ke layar yang sesuai dengan sisa pembagian yang diperoleh.

c. Struktur Kontrol Keputusan Bersarang

Sebuah struktur kontrol dapat menjadi bagian dari suatu struktur kontrol lain. Hal ini disebut *nested* atau tersarang. Pada contoh berikut, diberikan sebuah kode program yang memiliki struktur kontrol keputusan bersarang. Telusurilah program tersebut jika program diberi masukan 1000 dan 10.

```
/* Program dengan IF tersarang */

#include <stdio.h>
int main() {
    int a, b;
    scanf("%d %d", &a, &b);
    if (b!=0)
        if (a/b>10)
            printf("1\n");
    else
        printf("-1\n");
    return 0;
}
```

Ayo, Kita Berlatih 5: Latihan Struktur Kontrol Keputusan

Problem 1: Membagi Bilangan (Tingkat Kesulitan: ★ ★)

Buatlah sebuah program dari Diagram Alir 3: Membagi bilangan yang tersedia pada bagian algoritma di awal unit ini.

Problem 2: Bilangan Bulat Positif (Tingkat Kesulitan: ★ ★)

Deskripsi Soal:

Buatlah program untuk mengecek apakah sebuah bilangan bulat adalah bilangan bulat positif.

1. Jika bilangan bulat tersebut merupakan bilangan bulat positif, maka cetaklah “Bilangan Bulat Positif”.
2. Jika bilangan bulat tersebut bukan merupakan bilangan bulat positif, maka jangan cetak apa pun.

Format Masukan:

Sebuah bilangan bulat n . Nilai n berada pada rentang $-100 < n < 100$.

Format Keluaran:

Satu baris kalimat sesuai pada deskripsi soal

Contoh Kasus Uji

Masukan	Keluaran
3	Bilangan Bulat Positif
-10	

Problem 3: Jenis Bilangan Bulat (Tingkat Kesulitan: ★ ★ ★)

Deskripsi Soal:

Buatlah program untuk apakah sebuah bilangan bulat adalah bilangan termasuk bilangan bulat positif, negatif atau nol. Jika bilangan bulat tersebut merupakan bilangan bulat positif, cetak “Bilangan Bulat Positif”. Jika bilangan bulat tersebut merupakan bilangan bulat negatif, cetak “Bilangan Bulat Negatif”. Jika bilangan bulat tersebut merupakan bilangan bulat nol, cetak “Bilangan Bulat Nol”.

Format Masukan:

Sebuah bilangan bulat n . Nilai n berada pada rentang $-100 < n < 100$.

Format Keluaran:

Satu baris kalimat sesuai pada deskripsi soal.

Contoh Kasus Uji

Masukan	Keluaran
3	Bilangan Bulat Positif
-5	Bilangan Bulat Negatif

Problem 4: Nama Bulan (Tingkat Kesulitan: ★ ★ ★)

Deskripsi Soal:

Buatlah sebuah program yang menerima masukan bilangan bulat yang berada pada rentang 1 - 12, dan akan mencetak nama bulan yang sesuai dengan bilangan bulat tersebut. Apabila bilangan berada di luar rentang tersebut, cetak kalimat “Tidak ada bulan yang sesuai”.

Format Masukan:

Sebuah bilangan bulat n . Nilai n berada pada rentang $-100 < n < 100$.

Format Keluaran:

Satu baris kalimat sesuai pada deskripsi soal.

Contoh Kasus Uji

Masukan	Keluaran
1	Januari
3	Maret
13	Tidak ada bulan yang sesuai.

Problem 5: Mengecek Sisi Segitiga (Tingkat Kesulitan: ★ ★ ★ ★)

Tahukah kalian bahwa sebuah segitiga hanya bisa dibangun apabila sisi terpanjangnya lebih kecil daripada total panjang kedua sisi lainnya? Jika syarat ini tidak dipenuhi, tidak ada segitiga yang terbentuk.

Agria sedang membuat program untuk menghitung luas segitiga yang menerima masukan berupa tiga buah bilangan bulat yang merupakan panjang sisi segitiga tersebut. Akan tetapi, Agria menyadari bahwa ia harus terlebih dahulu memastikan ketiga panjang sisi yang dimasukkan benar-benar dapat membentuk sebuah segitiga. Oleh karena itu, ia merancang sebuah algoritma dalam bentuk diagram alir berikut untuk mengecek apakah ketiga bilangan tersebut dapat membentuk segitiga.

Tugas kalian adalah membantu Agria dengan membuat program berdasarkan diagram alir tersebut.

Problem 6: Belajar Membuat Kasus Uji (Tingkat Kesulitan: ★ ★ ★)

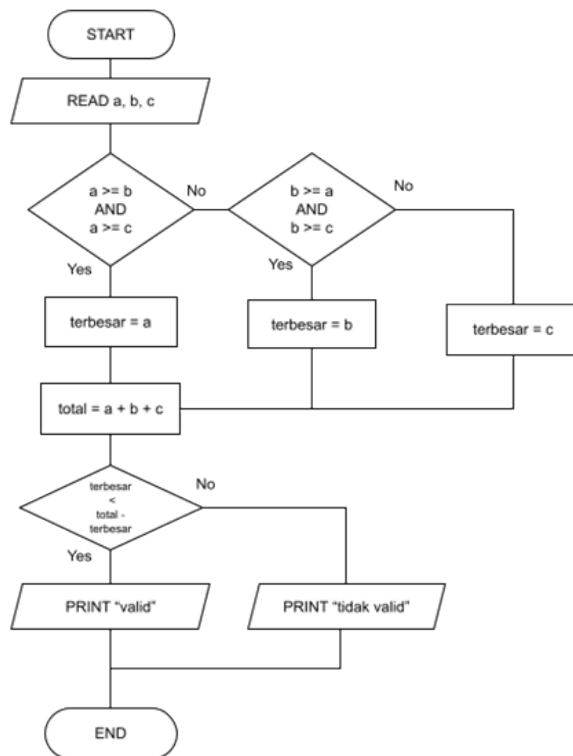
Perhatikan kembali diagram alir pada Problem 5. Diagram alir tersebut terlihat memiliki beberapa kemungkinan aliran, bergantung pada nilai masukan yang diberikan. Saat kalian mengecek program yang kalian buat dengan suatu kasus uji, kasus uji yang diberikan haruslah meliputi semua kemungkinan aliran tersebut. Sekarang, buatlah kasus uji sedemikian sehingga semua kemungkinan aliran pada diagram alir di atas dapat dicek.



Ayo Kita Renungkan

Jawablah pertanyaan berikut dalam Lembar Refleksi pada Buku Kerja. Jangan lupa mencatat kegiatan dalam Jurnal.

1. Pada bagian ini, kalian mendapatkan banyak konsep baru tentang program. Seperti apa perasaan kalian saat ini?
2. Apakah kalian bereksperimen dengan contoh-contoh yang diberikan di buku? Jika ya, pengetahuan paling menarik apa yang kalian temukan dari hasil eksperimen tersebut?
3. Dari latihan yang telah kalian kerjakan, terutama dari problem 1 dan 5, apa salah satu contoh penggunaan dari struktur kontrol keputusan?
4. Apakah kalian sudah memahami proses membuat kasus uji untuk menguji sebuah program, yang harus meliputi semua kemungkinan aliran pada diagram alir? Mengapa kalian harus memastikan semua kemungkinan aliran dicek?
5. Kesalahan apa yang sering kalian lakukan saat menulis kode program dengan menggunakan struktur kontrol keputusan?



6. Struktur Kontrol Perulangan

Kalian akan belajar memahami konsep dan cara kerja perulangan dalam pemrograman. Setelah menyelesaikan aktivitas ini, kalian diharapkan mampu menerapkan perulangan dan menghubungkannya dengan kejadian sehari-hari.

- a. Pernahkah kalian melakukan aktivitas yang sama berulang kali? Eits... tidak perlu jauh-jauh, untuk melangkah saja, kita mengulang pergerakan kaki kanan dan kiri. Apakah kalian selalu berpikir sebelum melakukan setiap langkah? Pertanyaan terakhir tersebut dapat ditanyakan pada semua kegiatan perulangan yang kalian lakukan.
- b. Membaca buku - kalian membalikkan halaman buku setiap selesai membaca halaman tersebut.

Salah satu keunggulan program komputer daripada manusia ialah kemampuannya untuk mengolah data yang berukuran besar atau melaksanakan suatu aksi berulang kali dalam periode waktu yang lama tanpa merasa bosan atau lelah. Hal ini dimungkinkan dengan adanya suatu kontrol perulangan. Pernyataan perulangan atau *loop* merupakan struktur program untuk keperluan iterasi, yaitu memproses satu atau beberapa pernyataan secara berulang (*looping*) berdasarkan kondisi tertentu. Program C menyediakan tiga bentuk pernyataan *loop*, yaitu:

1. `for` loop
2. `while` loop
3. `do...while` loop

a. Struktur Kontrol Perulangan `for`

Pernyataan ini umumnya digunakan untuk memproses pernyataan secara berulang-ulang, dengan jumlah perulangan yang dilakukan telah diketahui sebelumnya. Misalnya, berjalan sebanyak *n* langkah ke depan, atau mencetak barisan dari suku pertama hingga suku ke-*n*. Struktur kontrol perulangan `for` adalah sebagai berikut:

```
for (expr1; expr2; expr3)
{
    <statement>;
    ...
}
```

Struktur di atas akan dijalankan melalui proses berikut.

1. Ekspresi `expr1` akan dieksekusi ketika program menjalankan struktur `for` tersebut. Ekspresi ini biasanya berisi inisialisasi suatu variabel *counter* yang digunakan untuk menghitung jumlah perulangan yang telah dilakukan.
2. Ekspresi `expr2` merupakan suatu ekspresi bernilai benar atau salah (*boolean*) yang akan dicek sebelum pernyataan di dalam blok struktur dieksekusi. Apabila ekspresi ini bernilai benar, pernyataan akan dieksekusi. Sebaliknya, apabila ekspresi bernilai salah, pernyataan tidak akan dieksekusi dan perulangan berakhir. Dengan kata lain, pada bagian

ini, kalian menuliskan sebuah pernyataan yang merupakan kondisi berhenti (*stopping criteria*) untuk memastikan perulangan yang kalian buat memiliki langkah yang terbatas (dipastikan berhenti).

3. Ekspresi `expr3` merupakan sebuah pernyataan yang dijalankan setelah semua pernyataan di dalam struktur *for* dieksekusi. Biasanya, pernyataan ini dibuat untuk mengubah nilai variabel *counter* yang akan makin mendekati kondisi berhenti (memastikan nilai *counter* akan konvergen ke kondisi berhenti).

Walaupun ketiga ekspresi tersebut bersifat opsional (tidak harus ada), kalian disarankan untuk menuliskan ketiga ekspresi tersebut saat membuat program dengan jelas dan lengkap. Untuk saat ini, kalian perlu memahami teknik untuk menulis ketiga ekspresi tersebut dengan baik.

Perhatikan contoh kode program menulis bilangan bulat berikut. Pada contoh tersebut, program akan menulis bilangan bulat dari 0 hingga kurang dari *n*. Telusurilah kode program tersebut dengan teknik penelusuran yang telah kalian pelajari pada saat menelusuri suatu diagram alir!

```
/* Program Menulis Bilangan Bulat Sebanyak n Kali */  
  
#include <stdio.h>  
int main() {  
    int i, n;  
    scanf("%d", &n);  
    for (i=0; i < n; i=i+1) {  
        // i = i+1 dapat juga ditulis i++  
        printf("%d\n", i);  
    }  
    return 0;  
}
```

Keluaran program tersebut saat diberi masukan 5 adalah:

```
0  
1  
2  
3  
4
```

Pada contoh tersebut, variabel *counter* yang digunakan ialah *i* yang nilainya diinisialisasi (diberi nilai awal) dengan nilai 0. Pada bahasa C, lazimnya *counter*, dimulai dari nilai 0. Berbeda dengan proses pencacahan yang biasa kalian lakukan dari 1 di dunia nyata. Hal ini terkait dengan beberapa aspek teknis di bahasa C yang dimulai dari 0. Misalnya, di kelas XI nanti, kalian

akan menggunakan struktur data array yang dimulai dari indeks ke-0. Perlu diingat bahwa inisiasi dengan 0 ini hanyalah suatu kebiasaan masyarakat pemrogram dalam bahasa C, dan kalian tetap dapat melakukan pencacahan mulai dari 1. Kalian dapat mencoba mengubah kode program di atas sehingga *counter* berjalan dari 1. Selain *counter* yang berjalan menaik (*ascending*), kalian juga dapat membuat suatu *counter* yang berjalan turun (*descending*). Contoh ini disajikan misalnya pada Diagram Alir 4 dan 5 pada bagian algoritma.

Pernyataan yang ada pada *expr1* hingga *expr3* dapat ditulis menjadi deretan instruksi yang dipisahkan dengan tanda koma. Misalnya, terdapat pada contoh program berikut.

```
for (i=0, j=0; i<3; ++i, j++) {  
    printf("%d %d", i, j);  
}
```

Urutan pengerjaan akan sama seperti pada contoh sebelumnya. Akan tetapi, ada dua pernyataan yang akan dijalankan pada *expr1* hingga *expr3*.

Praktik Baik Pemrograman

Pada saat merancang sebuah struktur perulangan *for*, kalian perlu memastikan agar kondisi berhenti pasti akan tercapai (konvergen). Apabila kondisi berhenti tidak pernah tercapai, akibatnya, struktur ini akan berjalan terus-menerus dan menyebabkan terjadinya suatu perulangan yang tidak terbatas (*infinite loop*). Apabila hal ini terjadi, program akan dibekukan oleh sistem operasi, bahkan akan dihentikan. Perhatikan contoh berikut.

```
/* Program dengan Perulangan tak Terbatas */  
  
#include <stdio.h>  
  
int main()  
{  
    int i, n;  
    scanf("%d", &n);  
    for(i=0; i<n; i--)  
        printf("%d\n", n)  
    return 0;  
}
```

Dapat dilihat bahwa pada kode tersebut, nilai *counter* *i* akan berkurang dan tidak akan pernah melebihi nilai *n* jika *n* diisi dengan suatu bilangan bulat positif.



Lakukan penelusuran pada kode program tersebut dengan nilai $n = 3$.

b. Struktur Kontrol Perulangan While

Saat merancang perulangan, kalian bisa jadi tidak dapat menentukan berapa kali perulangan akan dilakukan. Akan tetapi, kalian mengetahui kondisi berhentinya. Misalkan instruksi berikut pada dunia nyata, “berjalan luruslah sampai ujung jalan, kemudian belok kiri.” Instruksi tersebut tidak memberikan gambaran jelas jumlah langkah yang akan kalian lakukan. Namun, secara naluriah, kalian mengetahui kapan kalian harus berhenti berjalan lurus, lalu berbelok ke arah kiri.

Pada program, suatu struktur kontrol *while* dikenal untuk melakukan perulangan seperti pada contoh di atas. Struktur kontrol tersebut dapat ditulis sebagai berikut. Pernyataan akan dieksekusi terus-menerus selama ekspresi kondisi bernilai benar.

```
while (ekspresi kondisi) {  
    <pernyataan>;  
    ...  
}
```

Sebagai contoh, misalnya kalian akan menulis kode program untuk membaca dan menuliskan kembali bilangan bulat positif. Hal ini terus dilakukan hingga program membaca nilai -1. Program tersebut dapat kalian lihat di bawah ini.

```
/* Program Baca Tulis Bilangan */  
  
#include <stdio.h>  
int main() {  
    int bilangan;  
    scanf("%d", &bilangan);  
    while (bilangan != -1) {  
        printf("%d\n", bilangan);  
        scanf("%d", &bilangan);  
    }  
    return 0;  
}
```



Lakukan penelusuran pada kode program ini dengan masukan yang diberikan ialah 1 2 3 4 -1.

c. Struktur Kontrol Perulangan Do - While

Struktur kontrol *do-while* memiliki perilaku yang mirip dengan *while*, yaitu kalian hanya mengetahui kondisi berhenti dari perulangan tersebut. Perbedaannya ialah struktur *do-while* dipastikan akan dikerjakan setidaknya satu kali. Bentuk umum pernyataan *do .. while* adalah sebagai berikut.

```
do {  
    <pernyataan>;  
} while (ekspresi kondisi);
```

Salah satu contoh penggunaan struktur *do-while* ialah ketika kalian menulis sebuah program interaktif yang akan meminta pengguna memasukkan kembali suatu nilai hingga nilai tersebut memenuhi suatu syarat. Hal ini akan sering kalian alami ketika kalian diminta untuk mengisi ulang (*retry*) saat menggunakan sebuah program atau mengisi sebuah formulir elektronik.

Misal, program berikut akan terus meminta pengguna memasukkan nilai sampai pengguna tersebut memasukkan bilangan bulat positif.

```
/* Program Verifikasi Masukan Pengguna */  
  
#include <stdio.h>  
int main() {  
    int bilangan;  
    do  
        scanf("%d\n", &bilangan);  
    while (! (bilangan > 0));  
    printf("Anda telah memasukkan bilangan bulat  
positif\n");  
    return 0;  
}
```

d. Struktur Kontrol Perulangan Bersarang

Sama seperti struktur kontrol keputusan, kalian dapat meletakkan struktur kontrol perulangan secara bersarang. Misalnya, pada contoh program berikut yang akan mencetak suatu pola berbentuk persegi menggunakan karakter asterisk '*'.


```

/* Program dengan FOR Bersarang */

#include <stdio.h>
const int PANJANG = 2;
const int TINGGI = 3;
int main() {
    int i, j;
    for (i = 0; i < TINGGI; i++) {
        for (j = 0; j < PANJANG; j++) {
            printf("*");
        }
        printf("\n");
    }
    return 0;
}

```



Ayo Cari Tahu

Lakukanlah penelusuran pada kode program tersebut!

Pada contoh di atas, *counter* pada struktur for terluar dan terdalam tidak saling berkaitan. Pada beberapa kasus nantinya, kedua *counter* tersebut bisa saja saling berkaitan. Misalnya, pada program berikut:

```

/* Program dengan Dua Counter */

#include <stdio.h>
int main() {
    int i, j, m, n;
    scanf("%d %d", &m, &n);
    for (i=1; i<=m; i++) {
        for (j=i; j<=n; j++) {
            printf("%d", j);
            if (j==n)
                printf("\n");
            else
                printf(" ");
        }
    }
    return 0;
}

```

Pada program tersebut, kalian dapat melihat bahwa nilai awal *counter* j akan berpengaruh pada nilai *counter* i. Saat dijalankan, keluaran dari program tersebut ialah:

```
1 2 3 4 5
2 3 4 5
3 4 5
```

Tentunya, tidak ada batasan jumlah struktur yang kalian buat secara bersarang. Kalian pun juga dapat memadukan struktur perulangan dengan struktur keputusan sehingga menghasilkan program yang lebih kompleks.

Ayo, Kita Berlatih 6: Latihan Struktur Kontrol Perulangan

Problem 1: Menghitung Mundur (Tingkat Kesulitan: ★ ★)

Buatlah kode program berdasarkan Diagram Alir 4 pada bagian algoritma untuk mencetak bilangan secara hitung mundur.

Problem 2: Menghitung Rataan (Tingkat Kesulitan: ★ ★ ★)

Deskripsi Soal:

Buatlah sebuah program yang akan menghitung rata-rata dari n buah bilangan.

Format Masukan:

Baris pertama berisi sebuah bilangan bulat positif n yang menunjukkan banyaknya data, sedangkan baris berikutnya berisi n buah bilangan bulat. Nilai n maksimal 1000, dan besarnya bilangan yang harus dihitung rata-ratanya berada pada rentang -1 miliar hingga 1 miliar.

Format Keluaran:

Nilai rata-rata dari n buah bilangan masukan. Nilai rata-rata tersebut dituliskan sebagai bilangan riil dengan dua angka di belakang titik desimal.

Contoh Kasus Uji

Masukan	Keluaran
10 20 30	20.00

Problem 3: Mencari Bilangan Terbesar (Tingkat Kesulitan: ★ ★ ★)

Buatlah kode program berdasarkan Diagram Alir 5 pada bagian algoritma untuk mencari bilangan terbesar dari sekumpulan bilangan yang diberikan.

Problem 4: Membuat Mesin Sortir Kembang Kol (Tingkat Kesulitan: ★ ★ ★ ★)

Deskripsi Soal:

Kalian akan membantu seorang petani kembang kol untuk menyortir kembang kol yang telah dipanen berdasarkan ukurannya. Kembang kol tersebut akan dikelompokkan menjadi berukuran kecil (< 50 gram per buah), berukuran sedang (50-200 gram per buah), dan berukuran jumbo (> 200 gram per buah). Selama ini, petani tersebut menyortir kembang kol menggunakan tenaga manusia. Karena kalian telah memiliki kemampuan untuk membuat program

yang dapat melakukan hal tersebut secara otomatis, kalian pun mengajak petani tersebut untuk menerapkan *future practice* dengan menerapkan otomatisasi. Bantulah petani tersebut dengan program buatan kalian! Ini akan menjadi langkah awal bagi petani tersebut untuk menjadi seorang petani modern.

Format Masukan:

Program kalian akan membaca berat dari setiap kembang kol dalam satuan gram. Berat setiap kembang kol ditulis dalam bilangan riil. Apabila semua kembang kol telah tersortir, program kalian akan membaca nilai -1.

Format Keluaran:

Tiga buah bilangan bulat yang merupakan jumlah kembang kol pada setiap kategori, yaitu kecil, sedang, dan jumbo. Setiap bilangan ditulis pada baris yang berbeda.

Contoh Kasus Uji

Masukan	Keluaran
100.0 20.5 300.1 40.1 -1	2 1 1

Problem 5: Memperbaiki Program (Tingkat Kesulitan: ★★★★★)

Deskripsi Soal:

Buatlah sebuah program untuk menggambar sebuah pola X dengan menggunakan karakter asterisk (*).

Format Masukan:

Sebuah bilangan bulat n yang menyatakan ukuran pola yang akan dibuat. Nilai n berada pada rentang 1 hingga 20.

Format Keluaran:

Pola huruf X yang sesuai dengan ukuran yang dimasukkan. Perhatikan perbedaan untuk banyak baris ganjil dan genap pada contoh keluaran.

Contoh Kasus Uji

Masukan	Keluaran
5	*_*_* _ *_ _ *_ _ *_ *_*_*
6	*_*_*_* _ *_* _ *_* _ *_* _ *_* *_*_**

Pada soal ini, kalian telah diberikan salah satu program yang dibuat untuk menyelesaikan permasalahan di atas. Akan tetapi, program tersebut tidak dapat dijalankan dan ditulis dengan tidak rapi! Pelajarilah kode program di bawah ini. Perbaiki program tersebut hingga dapat berjalan dengan benar. Sesuaikan pula penulisannya dengan mengikuti praktik baik yang telah kalian pelajari hingga saat ini.

```
int main() {
    int N, i, j;
    scanf("%d", &n);

    for(i=0; i>n; i++){
        for(j=0; j<n; j++){
            if(j==i || j==n-i-1) printf("-");
            else printf("*");
        }
        printf("\n");
    }
}
```



Ayo Kita Renungkan

Jawablah pertanyaan berikut dalam Lembar Refleksi pada Buku Kerja. Jangan lupa mencatat kegiatan dalam Jurnal.

1. Pada bagian ini, kalian mendapatkan banyak konsep baru tentang program. Seperti apa perasaan kalian saat ini?
2. Apakah kalian bereksperimen dengan contoh-contoh yang diberikan di buku? Jika ya, pengetahuan paling menarik apa yang kalian temukan dari hasil eksperimen tersebut?
3. Pada Problem 6, apakah kalian memikirkan dampak sosial yang terjadi pada pekerja yang selama ini menyortir kembang kol secara manual tersebut? Jika kalian menjadi petani tersebut, apa yang akan kalian lakukan kepada pekerja tersebut? Apa yang kalian lakukan jika kalian menjadi pekerja tersebut?
4. Kesalahan apa yang sering kalian lakukan saat membuat program menggunakan struktur kontrol perulangan?

7. Fungsi

Selama ini, kalian telah sering bertemu dengan fungsi. Di Matematika, kalian membuat suatu persamaan menggunakan fungsi seperti $y = f(x) = x + 1$. Fungsi $f(x)$ menerima sebuah masukan x yang disebut fungsi daerah dan menghasilkan

nilai *y* yang merupakan fungsi wilayah. Fungsi sangat berhubungan dengan kemampuan abstraksi yang telah kalian pelajari di berpikir komputasional sehingga program yang kalian tulis dapat ditulis dengan lebih baik.

Sejauh ini, kalian juga telah menggunakan beberapa fungsi dalam berlatih pemrograman. Pada Diagram Alir, kalian telah mengenal sebuah simbol subprogram untuk memberikan abstraksi dari suatu proses lain yang kalian gunakan dalam solusi kalian. Pada program bahasa C, struktur main merupakan sebuah fungsi yang akan dieksekusi oleh sistem operasi ketika program dijalankan. Selain itu, kalian pun telah menggunakan fungsi seperti `printf` dan `scanf`.

Pada hakikatnya, fungsi pada program melambangkan suatu kumpulan pernyataan yang memiliki tujuan tertentu. Tujuan tersebut direpresentasikan oleh nama dari fungsi tersebut. Misalnya, `scanf` yang memiliki fungsi untuk membaca (*scan*) nilai dari pengguna. Fungsi juga dapat menerima parameter-parameter, dan juga dapat mengembalikan suatu nilai. Dengan membungkus kumpulan instruksi tadi ke dalam suatu fungsi, kalian dapat menggunakan kembali fungsi tersebut di berbagai lokasi dalam program kalian.

a. Membuat Fungsi

Saat membuat suatu fungsi baru, kalian perlu menentukan tiga hal tersebut: nama fungsi yang merepresentasikan tujuan dari fungsi, parameter yang dimasukkan ke dalam fungsi, serta nilai yang dikembalikan. Ketiga informasi ini disebut prototipe dari fungsi. Adapun pernyataan-pernyataan yang ada di dalam fungsi tersebut disebut implementasi dari fungsi.

Misal, kalian akan membuat sebuah fungsi untuk menghitung luas lingkaran. Maka, kalian perlu menetapkan ketiga hal tersebut dan menghasilkan prototipe fungsi berikut. Dari prototipe tersebut, kalian dapat melihat bahwa fungsi `hitungLuasLingkaran` memerlukan sebuah parameter dengan tipe *data float* yang merupakan radius dari lingkaran. Saat dipanggil, fungsi ini akan mengembalikan sebuah nilai float yang merupakan luas lingkaran.

```
float hitungLuasLingkaran(float radius);
```

Jika dituliskan lengkap dengan implementasinya, fungsi tersebut dapat ditulis menjadi:

```
float hitungLuasLingkaran(float radius) {  
    float luas;  
    luas = 3.14 * radius * radius;  
    return luas;  
}
```

Pada kode program tersebut, kata kunci `return` digunakan untuk mengakhiri fungsi dan mengembalikan suatu nilai. Dalam hal ini, nilai yang dikembalikan ialah luas lingkaran.

Perlu diingat bahwa prototipe dari fungsi-fungsi berikut akan dianggap sebagai fungsi yang berbeda karena memiliki parameter yang berbeda dari tipe data. Hal ini disebut *overloading*.

```
float hitungLuasLingkaran(float radius);  
float hitungLuasLingkaran(int radius);
```

Kalian juga dapat membuat fungsi dengan jumlah parameter lebih dari satu. Misalnya, untuk menghitung luas persegi panjang berikut:

```
float hitungLuasPersegiPanjang(float panjang, float lebar);
```

Tentunya, kalian juga dapat membuat fungsi yang tidak memiliki parameter masukan, seperti yang kalian lakukan pada saat membuat fungsi `main()` pada program.

b. Memanggil Fungsi

Setelah dibuat, fungsi dapat dipanggil (function call) di dalam kode program. Perhatikan kode program berikut yang akan memanggil fungsi `hitungLuasLingkaran` yang telah dibuat.

```
/* Program Hitung Luas Bola */  
  
#include <stdio.h>  
float hitungLuasLingkaran(float radius) {  
    float luas;  
    luas = 3.14 * radius * radius;  
    return luas;  
}  
  
/* Program Hitung Luas Lingkaran */  
  
#include <stdio.h>  
float hitungLuasLingkaran(float radius) {  
    float luas;  
    luas = 3.14 * radius * radius;  
    return luas;  
}  
  
int main() {  
    float radius, luas;  
    scanf("%f", &radius);  
    luas = hitungLuasLingkaran(radius);  
    printf("%f\n", luas);  
}
```

Kalian juga dapat memanggil suatu fungsi di fungsi lain yang kalian buat. Misalnya, kalian ingin membuat fungsi untuk menghitung luas permukaan bola. Kalian dapat menulisnya menjadi:

```
float hitungLuasBola(float radius) {  
    float luas;  
    luas = 4.0 * hitungLuasLingkaran(radius);  
    return luas;  
}  
  
int main() {  
    float radius, luasBola;  
    scanf("%f", &radius);  
    luasBola = hitungLuasBola(radius);  
    printf("%f\n", luasBola);  
    return 0;  
}
```

c. Variabel Lokal pada Fungsi

Pada contoh-contoh di atas, kalian akan menemukan adanya deklarasi variabel dengan *identifier* yang sama. Misalnya, variabel *luas* kalian temukan di dalam fungsi *hitungLuasLingkaran* dan *hitungLuasBola*. Kedua variabel tersebut disimpan pada alamat memori yang berbeda yang hanya bisa diakses di dalam fungsi tempat variabel tersebut berada. Dengan kata lain, keduanya ialah variabel lokal yang tidak saling berhubungan.

Praktik Baik Pemrograman

Gunakan fungsi untuk melakukan abstraksi. Kumpulkan fungsi-fungsi yang telah kalian buat agar dapat digunakan kembali untuk membuat program dengan lebih cepat. Kumpulan fungsi ini dapat kalian satukan menjadi sebuah pustaka atau *library*. Apabila pustaka tersebut memiliki manfaat yang besar dan dibutuhkan oleh banyak orang, kalian dapat membuat pustaka tersebut menjadi publik.

Ayo, Kita Berlatih 7: Latihan Fungsi

1. Buatlah kode program dari Diagram Alir 1 pada bagian algoritma, yaitu menghitung luas permukaan kubus.
2. Buatlah sebuah fungsi untuk menghitung luas dan keliling bangun datar, seperti persegi panjang, lingkaran, dan segitiga.
3. Buatlah sebuah fungsi untuk menghitung luas permukaan bangun ruang seperti balok, kerucut, bola, dan limas.



Ayo Kita Renungkan

Jawablah pertanyaan berikut dalam Lembar Refleksi pada Buku Kerja. Jangan lupa mencatat kegiatan dalam Jurnal.

1. Pada bagian ini, kalian mendapatkan banyak konsep baru tentang program. Seperti apa perasaan kalian saat ini?
2. Apakah kalian bereksperimen dengan contoh-contoh yang diberikan di buku? Jika ya, pengetahuan paling menarik apa yang kalian temukan dari hasil eksperimen tersebut?
3. Apakah kalian telah merasakan manfaat dari penggunaan fungsi?
4. Kesalahan apa yang sering kalian lakukan saat membuat fungsi?

8. Penutup: Coding dan Pemrograman



(Sumber: Dokumen Kemendikbud,2021)

Pada bagian-bagian sebelumnya, kalian telah melaksanakan beberapa kegiatan latihan untuk mempelajari cara menuliskan program menggunakan bahasa C. Apakah ini berarti kalian telah menguasai pemrograman? Belum tentu karena kompetensi utama yang kalian pelajari ialah kemampuan menulis kode program yang disebut *coding*.

Pada Tabel 7.8 berikut, kalian dapat melihat perbedaan antara coding dan pemrograman dari berbagai aspek. Dengan memperhatikan tabel tersebut, kalian akan menyadari bahwa kemampuan yang kalian latih selama ini ialah kemampuan *coding*. Kalian juga dapat melihat hal-hal yang harus kalian pelajari lebih lanjut untuk menguasai pemrograman. Walaupun demikian, karena unit ini disusun dengan adanya aktivitas menyelesaikan permasalahan, kalian telah menyentuh beberapa aspek pemrograman seperti merancang algoritma, menguji

kode program, dan memberikan solusi sederhana pada suatu permasalahan yang diberikan. Perlu diingat juga, bahwa permasalahan diberikan pada kalian dalam bentuk spesifikasi program yang telah terstruktur. Pada pemrograman, kalian sendirilah yang harus merancang spesifikasi program tersebut.

Tabel 7.8 Perbedaan antara Coding dan Pemrograman

Aspek	Coding	Pemrograman
Definisi	Kegiatan menulis kode program dengan menggunakan bahasa pemrograman tertentu.	Proses mengembangkan sebuah solusi program untuk menyelesaikan masalah tertentu.
Perangkat yang Digunakan	Editor teks sudah cukup.	Perangkat analisis, desain, editor, dan pengujian.
Keahlian yang Dibutuhkan	Kemampuan membaca dan mengetik (menuliskan) sintaks bahasa pemrograman dengan benar.	Kemampuan berpikir komputasional, merancang algoritma, memodelkan permasalahan, dan manajemen proyek.
Hasil	Kode sederhana dengan banyak batasan.	Program yang siap digunakan oleh pengguna.

Pada unit PLB, kalian akan merasakan kegiatan pemrograman langsung untuk menyelesaikan permasalahan di sekitar kalian. Untuk mempersiapkan kalian mengikuti kegiatan PLB tersebut, di akhir unit ini, ada beberapa permasalahan yang diberikan untuk kalian kerjakan. Spesifikasi program tidak diberikan sehingga kalian perlu memanfaatkan kemampuan berpikir komputasional kalian dengan lebih maksimal.

Ayo, Kita Berlatih 8: Latihan Pemrograman

Pada bagian ini, kalian akan diberikan beberapa problem yang harus kalian selesaikan melalui pemrograman. Gunakan kemampuan berpikir komputasional kalian untuk menganalisis permasalahan yang diberikan. Rancanglah strategi dalam bentuk algoritma untuk menyelesaikan permasalahan tersebut. Kemudian, implementasikan algoritma tersebut ke dalam bentuk program menggunakan bahasa C.

Problem 1. Mencetak Faktor Prima (Tingkat Kesulitan: ★★★★★)

Kalian pasti telah mengenal faktor prima dari suatu bilangan. Misalnya, 20 akan memiliki faktor prima 2 dan 5. Rancang dan buatlah sebuah program untuk mencetak faktor prima dari sebuah bilangan bulat yang diberikan.

Problem 2. Mengecek Bilangan Prima (Tingkat Kesulitan: ★★★★★)

Apakah 713 merupakan bilangan prima? Rancang dan buatlah sebuah program yang dapat kalian gunakan untuk mengecek keprimaan suatu bilangan bulat yang diberikan.

Problem 3. Mengecek Tanggal (Tingkat Kesulitan: ★ ★ ★ ★)

Menulis tanggal tidak boleh sembarang. Ada aturan-aturan yang berlaku. Misalnya, ada aturan tahun kabisat serta jumlah hari dalam suatu bulan yang telah ditentukan. Pada problem ini, tugas kalian ialah merancang dan membuat sebuah program yang dapat memeriksa apakah sebuah tanggal dengan format yang kalian rancang, saat diberikan oleh pengguna dinyatakan benar.

D. Pengayaan

1. Menggunakan IDE Daring

Pada aktivitas ini, siswa dapat menggunakan kompilator atau *compiler* untuk membuat program. *Compiler* berfungsi untuk membaca dan menerjemahkan program yang kita tulis agar dapat dimengerti oleh komputer. Salah satu *online compiler* yang dapat siswa gunakan ialah *ideone* dengan cara mengunjungi situs <https://ideone.com/>. Berikut ini tampilan yang akan kalian lihat.

Skenario



Gambar 7.5 Tampilan Ideone

Sumber: Kemendikbud, 2021

Pada bagian bawah, terdapat beberapa menu yang dapat kita gunakan. Tombol paling ujung kiri berfungsi untuk memilih bahasa pemrograman yang akan digunakan. Terdapat berbagai macam bahasa pemrograman yang tersedia, tetapi yang akan digunakan di sini ialah bahasa C. Jadi, pastikan tombol paling ujung kiri bertuliskan “C”. Kemudian, di sebelahnya, terdapat tombol *stdin*. Tombol ini berfungsi untuk memasukkan *input* yang akan kita gunakan pada Aktivitas 3. Tombol berwarna hijau di ujung kanan berfungsi untuk menjalankan atau mengeksekusi program yang kita buat.

2. Berlatih Pemrograman Secara Mandiri Menggunakan Auto-Grader

Dalam dunia pemrograman, terutama dalam konteks berlatih atau berkompetisi, dikenal sebuah program yang dapat digunakan untuk mengecek apakah program yang kalian buat dapat menyelesaikan permasalahan yang diberikan. Program tersebut disebut *auto-grader* (penilai otomatis). Program tersebut lazim dimiliki oleh berbagai web tempat berlatih pemrograman untuk memudahkan murid memastikan program yang ia buat benar. Terdapat beberapa web yang dapat digunakan untuk berlatih pemrograman secara mandiri, salah satunya dikembangkan oleh Tim Olimpiade Komputer Indonesia yang dapat diakses di tlx.toki.or.id.

Web-web tersebut biasanya terdiri atas kumpulan materi dan soal yang dapat diselesaikan oleh kalian. Suatu soal, atau biasa disebut *problem*, terdiri atas deskripsi soal, format masukan, format keluaran, penjelasan, serta batasan yang digunakan dalam permasalahan tersebut. Setelah menganalisis dan memahami soal tersebut, kalian dapat merancang suatu algoritma untuk menyelesaikan permasalahan tersebut dalam batasan yang diberikan. Setelah itu, kalian dapat membuat kode program yang dapat digunakan untuk menyelesaikan soal tersebut dan mengirimkannya ke *auto-grader*.



Gambar 7.6 Contoh Soal yang Terdapat pada Web TLX yang Dikelola oleh Ikatan Alumni Tim Olimpiade Komputer Indonesia (IA-TOKI)

Sumber: Dokumen Kemendikbud, 2021

Auto-grader akan menjalankan program yang telah dibuat, kemudian memberikan suatu kumpulan masukan pada program tersebut. Hasil dari program dibandingkan dengan hasil yang benar, yang telah disimpan pada *auto-grader* oleh pembuat soal. Pasangan masukan-keluaran yang dimasukkan oleh pembuat soal ini disebut kasus uji (*test case*). Suatu soal bisa memiliki satu hingga banyak *test case* bergantung pada kompleksitas dan tingkat kesulitan soal tersebut.

Berdasarkan hasil pengujian tersebut, *auto-grader* akan memberikan skor pada program kalian. Skor diberikan berdasarkan persentase *test case* yang dijawab dengan benar oleh program yang kalian buat. Apabila program kalian menghasilkan lima jawaban benar dari sepuluh *test case* yang tersedia, kalian akan mendapatkan skor sebesar 50. Apabila semua *test case* dijawab dengan benar, kalian mendapatkan nilai 100 dan mendapatkan hasil ACCEPTED.

Selain hasil ACCEPTED, terdapat beberapa kemungkinan lain yang dikeluarkan oleh *auto-grader* pada program kalian. Misalnya, WRONG ANSWER diberikan apabila program kalian tidak mendapatkan nilai 100. TIME LIMIT EXCEEDED diberikan apabila program kalian berjalan lebih lama daripada yang diminta oleh pembuat soal. MEMORY LIMIT EXCEEDED diberikan apabila program kalian menggunakan memori yang lebih besar daripada yang diberikan pada soal. RUN TIME ERROR apabila program kalian berhenti secara tidak wajar saat dijalankan. COMPILER ERROR apabila kode program kalian gagal dikompilasi oleh *auto-grader*. Selain itu, masih terdapat hasil-hasil lainnya yang bergantung pada masalah yang terjadi pada saat kode program dijalankan.

Kalian dapat berlatih soal-soal yang tersedia pada kursus pemrograman dasar yang tersedia di TLX (tlx.toki.id/courses/basic). Kalian dapat membuat akun dan mulai mengikuti kursus tersebut dengan membaca materi dan mengerjakan soal-soal yang diberikan. Selain berupa modul, terkadang IA-TOKI juga mengadakan kontes pemrograman dengan berbagai tingkat kesulitan yang dapat kalian ikuti.

Setelah menamatkan materi pemrograman dasar, kalian dapat melanjutkan ke latihan-latihan yang lebih kompleks, yang dirancang untuk kompetisi-kompetisi pemrograman (tlx.toki.id/courses/competitive). Di tingkat SMA, ada International Olympiad Informatics (IOI), sedangkan di tingkat perguruan tinggi, terdapat International Collegiate Programming Contest (ICPC). Kedua kompetisi ini adalah sebuah gold-standard dari kompetisi pemrograman tingkat internasional.

Pemrograman merupakan kemampuan yang harus terus dilatih, dan tidak cukup dengan dihafal. Makin sering kita berlatih, dan semakin banyak soal yang kita kerjakan, kita akan makin cepat dan mudah mengerjakan soal dengan tingkat kesulitan yang lebih tinggi. Apabila kalian menemukan kesulitan, terdapat banyak forum serta media *online* dimana kalian dapat mencari petunjuk untuk membantu kalian.

3. Perbandingan Sintaks Bahasa C dan Python

Ada banyak bahasa pemrograman dengan kelebihan dan keunggulan masing-masing. Bahasa C pada unit ini hanyalah salah satu bahasa dari banyak bahasa pemrograman yang dapat dipilih. Pada tingkat SMP, serta pada unit analisis data di SMA Kelas X, diperkenalkan juga bahasa pemrograman lain, yaitu Python. Walaupun secara sintaks berbeda, struktur kontrol kedua bahasa tersebut mirip. Untuk mempermudah, diberikan Tabel 7.9 berikut yang memberikan sintaks pada bahasa C dan Python untuk mengerjakan suatu hal yang sama. Kalian dapat mempelajari lebih lanjut mengenai bahasa Python di python.org.

Tabel 7.9 Padanan Sintaks Bahasa C dan Bahasa Python

Sintaks pada Bahasa C	Sintaks pada Bahasa Python	Penjelasan
<code>printf("%d", x);</code>	<code>print(x)</code>	Mencetak keluaran
<code>scanf("%d", &x);</code>	<code>x = input()</code>	Membaca masukan
<code>x = 10;</code>	<code>x = 10</code>	Pemberian nilai
<code>a = (x + y)*z;</code>	<code>a = (x + y)*z</code>	Operasi aritmetika
<code>< <= > >= == !=</code>	<code>< <= > >= == !=</code>	Operasi perbandingan
<code>&&, </code>	<code>and , not</code>	
<code>if (<kondisi>) { pernyataan; }</code>	<code>if <kondisi>: pernyataan</code>	Percabangan dengan satu pernyataan
<code>if (<kondisi1>) { pernyataan1; } else if (<kondisi2>) { pernyataan2; } else { pernyataan3; }</code>	<code>if <kondisi1>: pernyataan1 elif <kondisi2>: pernyataan2 else: pernyataan3</code>	Percabangan dengan lebih dari satu kemungkinan pernyataan
<code>while (<kondisi>) { pernyataan; }</code>	<code>while <kondisi>: pernyataan</code>	Perulangan dengan while (berdasarkan kondisi)
<code>for (int i=0;i<n;i++) { pernyataan dengan i; }</code>	<code>for i in range(n): pernyataan dengan i</code>	Perulangan dengan iterator (variabel penunjuk)

Contoh Program: Menghitung Jumlah Bilangan dari 1 hingga N

Program dalam bahasa C:

```
/* Program Hitung Jumlah Bilangan 1 s/d N dalam Bahasa C */

#include <stdio.h>
int main() {
    int n, jumlah;
    scanf("%d", &n);
    jumlah = 0;
    for (int i = 0; i < n; i++)
        jumlah = jumlah + i;
    printf("jumlah bilangan 1 s/d %d = %d\n", n - 1,
    jumlah);
    return 0;
}
```

Program dalam bahasa Python:

```
# Program Hitung Jumlah Bilangan 1 - N di Python

n = input()
jumlah = 0;
for i in range(n):
    jumlah = jumlah + i
print("jumlah bilangan 1 s/d %d = %d\n" %( n - 1,
jumlah))
```

Menerjemahkan sebuah program dari satu bahasa ke bahasa lain itu mudah, bukan? Semudah kalian menerjemahkan “Saya cinta Indonesia” menjadi “I love Indonesia”!