KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
REPUBLIK INDONESIA
2021

Buku Panduan Guru
Informatika untuk SMA Kelas X

Penulis: Mushthofa, Dean Apriana Ramadhan, Auzi Asfarian, dkk.
ISBN: 978-602-244-502-9

Bab 7
Algoritma dan Pemrograman

Gambar 7.1Ilustrasi Unit Pembelajaran Algoritma dan Pemrograman
Sumber: Dokumen Kemendikbud, 2021

Di masyarakat, lebih dikenal istilah koding (*coding*) ketimbang pemrograman (*programming*). Dua istilah tersebut berbeda maknanya. *Coding* hanya merupakan *bagian dariprogramming*. Unit pembelajaran algoritma dan pemrograman ini diberikan untuk mengajarkan kemampuan membuat program dengan menggunakan bahasa pemrograman tertentu bagi siswa, bukan hanya koding. *Membuat program* **(***programming***)** pada hakikatnya ialah mengimplementasikan suatu strategi untuk menyelesaikan permasalahan tertentu ke dalam suatu bentuk yang dapat dipahami dan dieksekusi oleh komputer. Dengan demikian, *programming* membutuhkan kemampuan berpikir komputasional yang dimiliki oleh siswa untuk menyelesaikan suatu persoalan, yang solusinya ialah program komputer. Hal ini berlaku untuk semua bahasa pemrograman. Adapun aktivitas menulis *kode program (koding)* menitikberatkan pada menulis program menggunakan bahasa pemrograman tertentu. Koding makin dapat digantikan oleh generator kode dari hasil rancangan program. Belakangan ini, bahkan makin banyak perkakas pengembangan aplikasi dengan koding sesedikit mungkin yang disebut *low code development platform*. Sebaliknya, *programming* harus tetap dilakukan oleh manusia karena aspek ide solusi dan berpikirnya.

Setelah mempelajari pemrograman visual di jenjang SMP selama 3 tahun, dan mengalami transisi dari pemrograman visual ke bahasa tekstual di kelas IX, pada pelajaran pemrograman di kelas X ini, siswa akan belajar konsep koding dan pemrograman dengan menggunakan *bahasa pemrograman prosedural*. Bahasa yang dipilih pada buku ini ialah bahasa C. Ada banyak bahasa pemrograman lain, baik yang merupakan bagian dari paradigma prosedural atau paradigma lainnya, yang digunakan saat ini. Konsep-konsep inti yang diberikan dalam bahasa C, yaitu *variabel, ekspresi, struktur kontrol keputusan, dan struktur kontrolperulangan,* berlaku juga untuk bahasa pemrograman yang lain. Walaupun ada perbedaan sintaks dari sisi penulisan, konsep inti tersebut akan sama. Walaupun akan ada perbedaan antara bahasa pemrograman yang diberikan, ada *praktik baik*yang secara umum berlaku dalam pemrograman apapun, yang harus mulai dipupuk pada siswa sejak perkenalannya dengan bahasa pemrograman pertama.

Buku Siswa yang ditulis sebagai pasangan Buku Guru ini dirancang berbasis aktivitas agar siswa dapat berlatih dengan komputer dan dengan dukungan serta arahan dari guru. Siswa perlu diberi pemahaman bahwa pemrograman tidak mungkin dapat dikuasai hanya dengan menghafal sintaks, mencontoh program yang ada, atau mengetikkan sepotong kode orang lain. Siswa perlu banyak latihan agar dapat mencapai capaian pembelajaran yang telah ditetapkan. Buku Guru ini lebih banyak berisi penjelasan contoh program yang menjadi jawaban beserta dengan kasus uji yang dapat digunakan guru memeriksa program yang dibuat oleh siswa.

A. Tujuan Pembelajaran

Tujuan Pembelajaran untuk elemen Algoritma dan Pemrograman di kelas Xialah siswa mampu:

1. Membaca dan menulis algoritma dengan benar.
2. Memahami proses pemrograman dengan menggunakan bahasa pemrograman, pada buku ini dipilih bahasa C.
3. Memahami konsep variabel dan ekspresi dalam membuat program.
4. Memahami konsep struktur kontrol keputusan dan mengaplikasikan dalam bahasa C.
5. Memahami konsep struktur kontrol perulangan dan mengaplikasikan dalam bahasa C.
6. Memahami konsep fungsi dan implementasinya dalam bahasa C.
7. Memahami proses translasi dari satu bahasa ke bahasa lainnya melalui konsep yang sudah dikenalnya.

B. Kata Kunci

Penyelesaian persoalan (*problem solving*), algoritma, pemrograman, koding, *debugging, testing*

C. Kaitan dengan Bidang Pengetahuan Lain

Materi pada unit algoritma dan pemrograman ini berkaitan dengan unit-unit lain di bidang informatika. Dalam unit berpikir komputasional, siswa

diajarkan untuk menganalisis suatu permasalahan dan membuat strategi untuk menyelesaikan masalah tersebut. Strategi tersebut, lewat unit ini, diimplementasikan dalam bentuk program yang dapat dijalankan oleh komputer. Dengan demikian, berpikir komputasional harus dikuasai oleh siswa untuk menghasilkan suatu program yang merupakan solusi dari masalah yang akan diselesaikan.

Literasi komputer yang diberikan melalui materi TIK menjadi syarat perlu bagi siswa saat menggunakan komputer untuk menulis program. Ketika program dijalankan, sistem komputer telah didesain sedemikian rupa sehingga program mampu mengakses layanan-layanan yang diberikan oleh sistem operasi, misalnya seperti alokasi memori yang digunakan oleh program yang berjalan. Kemampuan pemrograman juga memiliki peran penting dalam melakukan analisis data menggunakan komputer.

Pemrograman dibutuhkan pada unit PLB karena pada PLB, siswa diharapkan menghasilkan solusi berupa program komputer.Selain di bidang informatika, kemampuan pemrograman pada saat ini juga dimanfaatkan pada banyak bidang ilmu, bahkan melahirkan cabang keilmuan seperti bioinformatika, geoinformatika, informatika kesehatan, dan bidang multidisiplin lainnya.

D. Strategi Pembelajaran

Belajar pemrograman berarti banyak berlatih. Oleh karena itu, pada unit ini, siswa sangat dianjurkan untuk belajar pemrograman dengan banyak berlatih. Guru dapat memberikan latihan yang ada di Buku Siswa, dan perlu memberikan waktu yang cukup bagi siswa untuk mengerjakan aktivitas yang diberikan. Pada saat mengerjakan aktivitas, guru berperan sebagai mentor yang membantu siswa dalam mengembangkan program. Guru disarankan untuk meminta siswa yang berhasil mengerjakan untuk menjelaskan ke teman dan saling berbagi kesulitan yang dihadapi.

Menjaga semangat dan memotivasi siswa dalam mempelajari pemrograman sangatlah penting. Siswa perlu diberi pemahaman bahwa belajar pemrograman bukanlah menghafal, mengetik kode, tetapi latihan berpikir. Ketika baru pertama kali berlatih, akan ada banyak kesulitan dan kebingungan, dan itu

hal yang wajar. Saat menemukan suatu jenis permasalahan baru, atau suatu elemen pemrograman baru, pengalaman siswa masih sedikit. Dari sisi berpikir komputasional, hal ini berarti kemampuan siswa untuk mengenali pola belum dapat digunakan secara maksimal. Seiring dengan makin banyak problema yang diselesaikan oleh siswa, pengalaman ini akan terakumulasi dan membuat siswa dapat menyelesaikan problem serupa dengan waktu yang lebih cepat.

Berdasarkan pengalaman, kemampuan siswa sangat beragam karena minat dan motivasi. Selain itu, akses pada komputer yang dibutuhkan untuk berlatih pun akan memiliki variasi. Oleh karena itu, guru perlu menyesuaikan strategi pembelajaran di kelas dengan kondisi yang ada di kelas tersebut. Pada beberapa pertemuan, waktu dibuat cukup longgar sehingga memungkinkan adanya ruang bagi siswa untuk mengatasi kesulitannya. Akan tetapi, jika kondisi di lapangan memerlukan waktu lebih lama, guru dapat membatasi jumlah problem yang diberikan.

1. Laboratorium Komputer (Plugged) atau Kelas (Unplugged)

Ada beberapa strategi yang dapat digunakan oleh guru dalam mengelola kelas pemrograman, terutama dengan mempertimbangkan akses pada sarana komputer yang digunakan dalam belajar pemrograman.

1. *Strategi pertama: berlatih di komputer lokal*. Pada strategi ini, siswa akan melakukan pemrograman dengan komputer, baik yang mereka miliki atau disediakan di sekolah. *Tools* yang dipakai dapat diinstalasi di komputer, atau memanfaatkan kompiler *online*. Kode program kemudian dinilai oleh manusia, baik oleh guru maupun diperiksa secara bergantian oleh siswa lainnya menggunakan himpunan kasus uji yang diberikan. Cara ini dilakukan apabila sekolah telah memiliki sarana lab komputer atau siswa dapat membawa komputernya masing-masing. Apabila jumlah komputer terbatas, siswa dapat mengerjakan secara berkelompok.
2. *Strategi kedua: berlatih dengan auto-grader*. Latihan dengan autograder membutuhkan koneksi internet. Pada strategi ini, siswa dapat mengirimkan kode program mereka ke *auto-grader* yang telah disiapkan. *Auto-grader* akan menampilkan problem-problem yang terdapat pada Buku Siswa. Siswa kemudian dapat memilih problem yang sesuai, kemudian ia dapat

mengirimkan kode jawaban mereka ke situs *auto-grader*. *Auto-grader* akan membaca dan menjalankan program tersebut dan secara otomatis akan mengecek kebenaran program tersebut dengan kasus uji yang telah tersedia. *Auto-grader* kemudian akan memberikan nilai pada program tersebut berdasarkan jumlah kasus uji yang dijawab dengan benar. Apabila program dibuat dengan salah, *auto-grader* akan menampilkan pesan kesalahan. Materi ini dapat dilihat lebih lanjut pada bagian pengayaan Buku Siswa: Berlatih Pemrograman Secara Mandiri Menggunakan Auto-Grader.

3. *Strategi ketiga: menyimulasikanprogram*. Strategi ini diberikan dalam kondisi perangkat komputer yang sangat sedikit atau tidak tersedia bagi siswa. Pada strategi ini, guru dapat menyesuaikan aktivitas menjadi kegiatan menuliskan algoritma, baik dalam bentuk diagram alir, deskripsi tingkat tinggi, atau *pseudocode* dari setiap problem yang diberikan.

2. Aktivitas Belajar Berpasangan

Berlatih pemrograman mungkin sulit bagi sebagian siswa. Di sisi lain, akan ada siswa yang menguasai dengan cepat, dan ada yang membutuhkan bantuan lebih. Selain itu, membuat program membutuhkan ketelitian, yang terkadang sulit dilakukan oleh siswa terhadap programnya sendiri. Oleh karena itu, salah satu strategi yang dapat digunakan ialah aktivitas pemrograman secara berpasangan (*pair programming*). Pemrograman berpasangan ini, sesuai namanya, dilakukan dengan memasangkan dua orang siswa di komputer yang sama. Terdapat dua peran, yaitu *driver* dan *navigator*. *Driver* ialah siswa yang membuat program, sedangkan *navigator*ialah siswa yang akan memeriksa program yang dibuat oleh siswa yang berperan sebagai *driver* dan memberikan umpan balik atau arahan kepadanya. Siswa kemudian saling bertukar peran hingga keduanya dapat menyelesaikan permasalahan yang diberikan. Proses saling cek ini akan sangat membantu guru untuk memastikan setiap siswa mendapatkan konstruksi umpan balik yang mereka perlukan untuk menghasilkan program yang dapat berjalan dengan benar.

Gambar 7.2 Ilustrasi Aktivitas Berpasangan
Sumber: Wikimedia Commons (https://upload.wikimedia.org/wikipedia/commons/thumb/a/af/Pair_programming_1.jpg/330px-Pair_programming_1.jpg)

3. Belajar Menelusuri Program

Belajar algoritma dan pemrograman menuntut siswa untuk dapat menyimulasikan suatu algoritma atau kode program. Hal ini disebut sebagai kemampuan menelusuri program (*tracing*). Saat melakukan penelusuran, siswa berperan sebagai komputer yang akan membaca setiap instruksi satu per satu, mengeksekusinya secara manual, dan mencatat masukan, hasil suatu proses, dan keluaran dari algoritma ada kode program di lembar kerjanya. Hal ini sangat mirip dengan suatu komputer yang akan menjalankan program dan menggunakan memori untuk menyimpan data.

Penelusuran ini dapat dilakukan tanpa menggunakan komputer (*unplugged*). Salah satu cara untuk menelusuri telah diberikan pada Buku Siswa. Dengan menelusuri, siswa dapat memahami cara kerja suatu algoritma atau kode program. Selain itu, siswa pun dapat menggunakan teknik ini untuk mencari kesalahan yang terjadi, atau disebut *debugging*. Walaupun contoh diberikan pada suatu diagram alir, teknik tersebut juga dapat digunakan untuk *pseudocode* atau kode program.

Kegiatan menelusuri program dapat dilakukan untuk menelusuri contoh kode program yang disediakan pada Buku Siswa, kode program yang dibuat oleh siswa lainnya (siswa saling menelusuri), atau contoh kode program yang diberikan oleh guru. Di Buku Siswa, terdapat juga beberapa soal yang diberikan sebagai latihan untuk membaca, mengidentifikasi kesalahan, dan memperbaiki kode program.

4. Praktik Baik Pemrograman

Pada setiap bab di Buku Siswa, praktik baik pemrograman diberikan secara bertahap seiring dengan kemajuan belajar siswa. Praktik baik ini sangat perlu diperkenalkan kepada siswa sejak awal mempelajari pemrograman untuk membentuk kebiasaan baik dalam membuat program. Praktik baik ini akan membantu siswa untuk lebih disiplin dan menghasilkan kode program yang berkualitas tinggi: mudah dibaca dan dipahami serta berjalan dengan benar. Oleh karena itu, guru sangat diharapkan untuk memberikan perhatian terhadap penerapan praktik baik pemrograman ini, dan memberikan umpan balik terhadap praktik yang kurang baik dengan mengoreksinya, atau mengapresiasi jika dilakukan.

5. Asesmen dan Integritas

Perlu diingat bahwa kode program yang diberikan pada pembahasan di Buku Guru ini hanyalah salah satu contoh dari program yang benar. Pada kenyataannya, kode program yang dibuat oleh siswa akan sangat bervariasi. Sangat perlu diperhatikan bagi guru untuk tidak hanya memeriksa hasil kode, apalagi dengan hanya membandingkan secara persis sama dengan kode yang diberikan pada buku ini.

Sebagai gantinya, yang perlu dicek oleh guru ialah *proses pembuatan program, kualitas penulisan program, perilaku dari program, dan kejujuran siswa dalam mengerjakan*. Proses pembuatan program berarti langkah-langkah yang dilalui oleh siswa untuk menghasilkan program tersebut. Kualitas penulisan program berarti kode program ditulis mengikuti standar praktik baik pemrograman sehingga terstruktur, rapi, dan mudah untuk dibaca. Terakhir, perilaku dari program berarti program memberikan hasil yang benar apabila dijalankan.

Integritas sangat penting dalam menguasai pemrograman. Siswa yang melakukan plagiasi, atau menyalin kode program dari teman sekelasnya atau sumber lain di internet, akan kehilangan proses berpikir yang diharapkan terbentuk dari pembelajaran pada unit ini. Pemrograman merupakan sebuah kemahiran yang harus diasah dengan berlatih. Selain itu, menyalin kode merupakan sebuah praktik buruk dalam pemrograman yang perlu dihindari sejak awal melakukan kegiatan pemrograman.

E. Organisasi Pembelajaran

Bab ini dibagi menjadi dua bagian, yaitu mengenal algoritma dan pemrograman serta membuat program dengan menggunakan bahasa C. Bagian pertama diperkenalkan untuk memberikan kemampuan dasar bagi siswa untuk mulai membuat program, yang meliputi kemampuan membaca dan menulis algoritma, kemampuan mempersiapkan lingkungan pemrograman, serta kemampuan untuk menulis dan menjalankan program sederhana yang dapat menerima masukan (*input*) dan menghasilkan suatu luaran (*output*). Bagian kedua merupakan pengenalan terhadap struktur-struktur kontrol dalam pemrograman yang menjadi fondasi siswa dalam membuat program yang lebih kompleks. Pada bagian kedua ini, siswa harus membuat program untuk menyelesaikan suatu permasalahan yang diberikan.

Perlu dicatat oleh guru bahwa sangat perlu bagi siswa untuk menguasai bagian pertama sebelum mereka melaksanakan bagian kedua. Bagian kedua pun demikian, sangat penting agar siswa dapat berlatih dengan benar dan menguasai bagian sebelumnya agar dapat memperoleh hasil maksimal di bagian berikutnya.

Tabel 7.1Organisasi Pembelajaran Unit Algoritma dan Pemrograman

Materi	Durasi (JP)	Tujuan Pembelajaran	Aktivitas
Mengenal Algoritma dan Pemrograman	1 JP	Siswa mampu membaca dan menulis algoritma dengan benar.	Ayo Berlatih 1: Latihan Menelusuri Diagram Alir
	2 JP	Siswa mampu membaca dan menulis algoritma dengan benar.	Ayo Berlatih 2: Latihan Menulis Algoritma
	1 JP	Siswa mampu menginstal ***tools*** dan lingkungan pemrograman yang akan dipakai berlatih. (***Aktivitas ini bersifat opsional jika tools belum tersedia di laboratorium komputer atau jika siswa akan menginstal di komputer pribadinya.*** ***Jika materi akan dilewati, bisa langsung menuju ke pertemuan 4 dengan penambahan latihan contoh program untuk diketik ulang dan dimodifikasi.***)	Ayo, Lakukan 1: Instalasi IDE Bahasa C
	2 JP	Siswa mampu mengetik ulang kode program dan menjalankannya dalam lingkungan pemrograman yang dipergunakan.	Ayo, Lakukan 2: Membuat Program Pertama dengan Bahasa C
	3 JP	Siswa memahami konsep input-output dan mampu menuliskan program sederhana yang membaca dan menulis.	Ayo, Berlatih 3: Latihan Input-Output
Membuat Program dengan Bahasa C	3 JP	Siswa memahami konsep variabel dan ekspresi dan mengaplikasikannya dalam bentuk program.	Ayo, Berlatih 4: Latihan Ekspresi
	3 JP	Siswa memahami konsep struktur kontrol keputusan dan mengaplikasikannya dalam bentuk program.	Ayo, Berlatih 5: Latihan Struktur Kontrol Keputusan
	3 JP	Siswa memahami konsep struktur kontrol perulangan dan mengaplikasikannya dalam bentuk program.	Ayo, Berlatih 6: Latihan Struktur Kontrol Perulangan
	3 JP	Siswa memahami implementasi fungsi dalam program.	Ayo, Berlatih 7: Latihan Fungsi
	6 JP	Siswa mampu mengaplikasikan konsep-konsep pemrograman prosedural (baca tulis, variabel, ekspresi, struktur kontrol keputusan dan pengulangan, serta fungsi) dalam menyelesaikan persoalan yang lebih kompleks.	Ayo, Berlatih 8: Latihan Pemrograman

Selain materi di atas, diberikan pula tiga topik pengayaan, yaitu (1) Menggunakan IDE Daring, (2) Berlatih Pemrograman Secara Mandiri Menggunakan *Auto-Grader*, dan (3) Perbandingan Sintaks Bahasa C dengan Python.

F. Pengalaman Belajar Bermakna, Profil Pelajar Pancasia, Berpikir Komputasional dan Praktik Inti

Tabel 7.2Pengalaman Bermakna, Profil Pelajar Pancasila, Praktik Inti, dan Berpikir Komputasional UnitAP

Pengalaman Bermakna	Profil Pelajar Pancasila	Berpikir Komputasional	Praktik Inti
Mengenal Algoritma dan Pemrograman	Mandiri, Bernalar Kritis	Abstraksi, Algoritma, Dekomposisi, dan Pengenalan pola	Memahami perangkat, mengimplemen-tasikan perangkat yang sesuai.
Membuat Program dengan Bahasa C	Mandiri, Bernalar Kritis	Abstraksi, Algoritma, Dekomposisi, dan Pengenalan pola. Penyelesaian persoalan sederhana.	Menyelesaikan masalah dan mengimplemen-tasikan program.

G. Panduan Pembelajaran

1. Pertemuan 1: Mengenal Algoritma dan Pemrograman (1 JP)

Tujuan Pembelajaran:

Siswa mampu membaca dan menulis algoritma dengan benar.

Apersepsi

Pada kehidupan sehari-hari, ada banyak langkah terstruktur dan telah terdefinisi dengan baik yang dapat menjadi petunjuk bagi manusia untuk mencapai tujuan tertentu. Dalam membuat program, langkah-langkah terstruktur dan terdefinisi dengan baik ini juga diperlukan agar komputer dapat bekerja dengan baik. Oleh karena itu, pada pertemuan kali ini, siswa akan mulai belajar algoritma dan pemrograman dengan membaca beberapa diagram alir yang telah disediakan, dan berlatih melakukan penelusuran diagram ali tersebut.

Pemanasan

Guru disarankan untuk membawa peraga atau foto peraga dari langkah-langkah terstruktur di dunia nyata. Beberapa contoh yang paling mudah ialah resep

memasak atau prosedur administrasi di sekolah (misal prosedur membayar uang sekolah).

Kebutuhan Sarana dan Prasarana

Aktivitas ini tidak memerlukan komputer dan bersifat *unplugged*.

Kegiatan Inti

1. (5 Menit) Guru membuka kelas dan dapat melakukan pemanasan.
2. (10 Menit) Guru mengenalkan notasi diagram alir dan memberikan contoh cara menelusurinya.
3. (25 Menit) Guru mengarahkan siswa ke aktivitas **AP-K10-01-U**: Latihan Menelusuri Diagram Alir. Berikan waktu kepada siswa untuk bekerja mandiri (atau berkelompok) untuk melakukan penelusuran di buku mereka. Beberapa soal dapat diberikan sebagai pekerjaan rumah.
4. (5 Menit) Guru menutup kelas dan mengarahkan siswa pada aktivitas refleksi. Hasil pengerjaan diagram alir dapat dikumpulkan untuk dinilai.

Aspek Kreativitas

Guru dapat mengambil kasus lain yang lebih dekat dengan siswa. Ambil contoh-contoh terkini yang sebisa mungkin prosesnya telah dipahami oleh siswa sehingga siswa dapat menggunakan waktu secara maksimal untuk menyusun diagram alir yang tepat dan dapat dipahami.

2. Pertemuan 2: Mengenal Algoritma dan Pemrograman (2 JP)

Tujuan Pembelajaran:

Siswa mampu membaca dan menulis algoritma dengan benar.

Apersepsi

Setelah membaca dan menelusuri, sekarang saatnya untuk menulis suatu algoritma. Menulis algoritma tidak boleh dilakukan sembarangan. Saat menulis, siswa perlu memastikan algoritma yang mereka tulis dapat dipahami oleh orang lain dengan mudah, dan instruksi-instruksinya terdefinisi dengan baik sehingga dapat diimplementasikan dalam bentuk program dengan mudah.

Pemanasan

Guru disarankan untuk menampilkan sebuah algoritma sederhana dan menanyakan pada siswa proses apa yang direpresentasikan oleh algoritma tersebut.

Kebutuhan Sarana dan Prasarana

Aktivitas ini tidak memerlukan komputer dan bersifat *unplugged*.

Kegiatan Inti

1. (5 Menit) Guru membuka kelas dan dapat melakukan pemanasan.
2. (15 Menit) Guru memberikan contoh pembuatan algoritma dari sebuah operasi matematika yang telah dikenal oleh siswa (misal: menghitung bangun datar atau sejenisnya).
3. (40 Menit) Guru mengarahkan siswa ke aktivitas **AP-K10-02-U**: Menulis Algoritma. Berikan waktu kepada siswa untuk bekerja mandiri (atau berkelompok) untuk menulis algoritma, baik dalam bentuk diagram alir atau pseudokode, pada buku mereka.
4. (25 Menit) Guru meminta siswa saling bertukar hasil pekerjaan, dan menelusurinya. Pancing siswa untuk memberikan umpan balik konstruktif satu sama lain.
5. (5 Menit) Guru menutup kelas dan mengarahkan siswa pada aktivitas refleksi. Hasil pengerjaan diagram alir dapat dikumpulkan untuk dinilai.

Aspek Kreativitas

Guru dapat mengambil kasus lain yang lebih dekat dengan siswa. Ambil contoh-contoh terkini yang sebisa mungkin prosesnya telah dipahami oleh siswa sehingga siswa dapat menggunakan waktu secara maksimal untuk menyusun diagram alir yang tepat dan dapat dipahami.

Pembahasan Ayo, Kita Berlatih 2

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

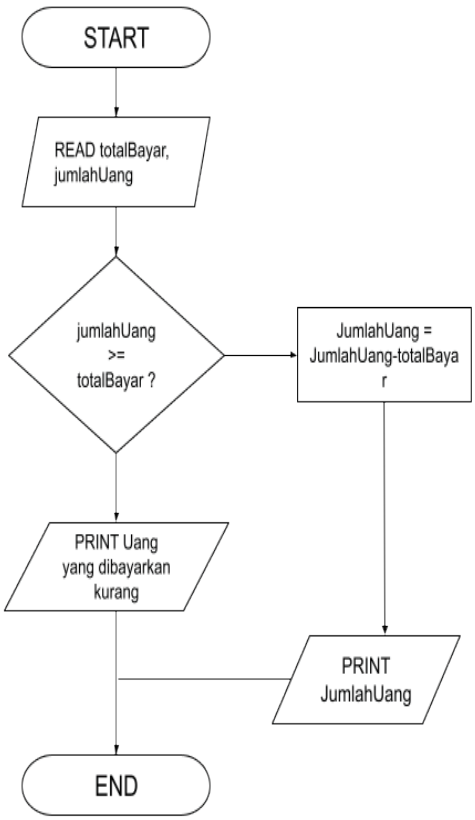
Penjelasan singkat:

Masukan atau input program ialah total bayar dan jumlah uang yang dibayarkan pelanggan.

Kemudian, program akan melakukan pengecekan terhadap jumlah uang yang dibayarkan.

Kondisi 1 : Jumlah uang >= Total bayar
Keluaran 1 : Jumlah uang - Total bayar
Kondisi 2 : Jumlah uang < Total bayar
Keluaran 2 : Uang yang dibayarkan kurang

Solusi Diagram Alir:



Gambar 7.3 Flowchart Soal Membayar Bakso
Sumber: Dokumen Kemendikbud, 2021

Solusi Pseudokode:

Deskripsi Tingkat Tinggi	Pseudokode

• Baca total bayar dan jumlah uang yang dibayarkan.
• Jika jumlah uang yang dibayarkan lebih besar atau sama dengan total bayar, kurangi jumlah uang sebesar total bayar, lalu cetak jumlah uang.
• Jika jumlah uang yang dibayarkan lebih kecil dari total bayar, cetak kalimat 'Uang yang dibayarkan kurang'.
	Algoritma membayar bakso
	Input: Total bayar dan Jumlah uang yang dibayarkan pelanggan
	
	Input totalBayar, jumlahUang
	if totalBayar >= jumlahUang
	  jumlahUang = jumlahUang-totalBayar
	  Print jumlahUang
	Else
	  Print "Uang yang dibayarkan kurang"

Contoh Kasus Uji:

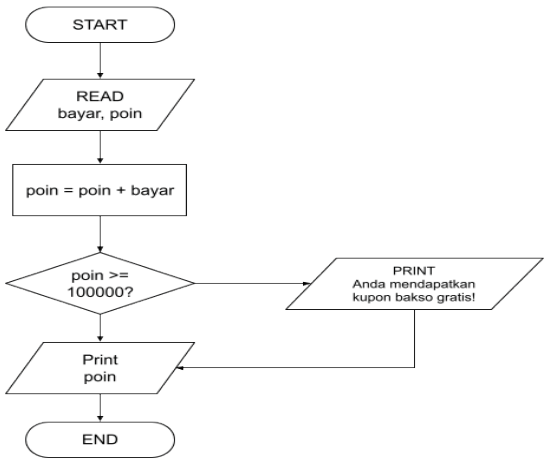
No.	Masukan	Keluaran	Keterangan
1	1 2	1 2	contoh kasus yang ada di soal
2	23 56	23 56	contoh kasus uji yang ada di tengah jangkauan batasan input

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

Penjelasan singkat:

1. Masukan atau input terdiri atas total pembayaran dan total poin pelanggan saat ini.
2. Program akan menjumlahkan poin pelanggan saat ini sejumlah total pembayaran.
3. Program akan mengecek jika poin pelanggan lebih dari sama dengan 100.000,akan mencetak kalimat 'Anda mendapatkan kupon bakso gratis!'.
4. Program mencetak total poin pelanggan saat ini.

Solusi Diagram Alir:



Gambar 7.4 Flowchart Soal Hadiah Bakso Gratis
Sumber: Dokumen Kemendikbud, 2021

Solusi Pseudokode:

Deskripsi tingkat tinggi	Pseudokode
• Baca total pembayaran dan total poin pelanggan saat ini. • Jumlahkan total poin pelanggan saat ini sebesar total pembayaran. • Jika total poin pelanggan lebih besar atau sama dengan 100.000, cetak 'Anda mendapatkan kupon bakso gratis'. • Cetak total poin pelanggan saat ini.	Algoritma hadiah bakso gratis Input: Total pembayaran dan total poin pelanggan saat ini (bayar dan poin) Input bayar, poin poin = poin + bayar if (poin >= 100000) Print "Anda mendapatkan kupon bakso gratis" Print poin

Contoh kasus uji

Kasus	Masukan	Keluaran
1	Total Pembayaran: 80.000 Total Poin Pelanggan Saat Ini: 10.000	Poin Anda saat ini: 90000
2	Total Pembayaran: 20.000 Total Poin Pelanggan Saat Ini: 90.000	Anda mendapatkan kupon bakso gratis! Poin Anda saat ini: 110000

3. Pertemuan 3: Mengenal Algoritma dan Pemrograman (1 JP)

Tujuan Pembelajaran:

Siswa mampu menginstal *tools* dan lingkungan pemrograman yang akan dipakai berlatih.

(Aktivitas ini bersifat opsional jika tools belum tersedia di laboratorium komputer atau jika siswa akan menginstal di komputer pribadinya. Jika materi akan dilewati, bisa langsung menuju ke pertemuan 4 dengan penambahan latihan contoh program untuk diketik ulang dan dimodifikasi.)

Apersepsi

Saatnya mulai membuat program. Akan tetapi, sebelum mulai, lingkungan untuk bekerja perlu disiapkan terlebih dahulu. Lingkungan ini merupakan sekumpulan perangkat lunak untuk membantu siswa dalam membuat program. Pada aktivitas ini, siswa dituntut untuk melakukan instalasi lingkungan pengembangan pemrograman, dan belajar menggunakannya untuk membuat suatu program sederhana.

Pemanasan

Guru disarankan untuk memberikan contoh singkat penggunaan lingkungan pemrograman tersebut dan menjalankan sebuah program sederhana.

Kebutuhan Sarana dan Prasarana

Aktivitas ini memerlukan komputer. Apabila koneksi internet tidak tersedia, guru perlu mengunduh file IDE yang akan diinstal terlebih dahulu dan mendistribusikan berkas instalasi dengan cara lain pada siswa.

Kegiatan Inti

1. (5 Menit) Guru membuka kelas dan dapat melakukan pemanasan.
2. (5 Menit) Guru mendistribusikan berkas instalasi kepada siswa.
3. (20 Menit) Guru mengarahkan siswa ke aktivitas **AP-K10-03**: Instalasi IDE Bahasa C. Penting: mintalah siswa untuk membaca dengan saksama setiap tampilan dari sistem, dan mengikuti petunjuk yang sesuai dengan sistem operasi yang digunakan.

4. (10 Menit) Minta siswa mengetikkan kode program yang ada pada buku, dan menjalankannya.
5. (5 Menit) Guru menutup kelas dan mengarahkan siswa pada aktivitas refleksi. Minta siswa untuk melakukan instalasi di komputer yang dapat mereka.

4. Pertemuan 4: Mengenal Algoritma dan Pemrograman (2 JP)

Tujuan Pembelajaran:

Siswa mampu mengetik ulang kode program dan menjalankannya dalam lingkungan pemrograman yang dipergunakan.

Apersepsi

Saatnya membuat program. Pada pertemuan kali ini, siswa dapat mengetik ulang (bukan salin tempel) kode program yang diberikan untuk merasakan pengalaman pertama dalam menulis kode program menggunakan bahasa C. Waktu aktivitas diberikan cukup panjang untuk memberikan waktu bagi siswa untuk mengatasi kesulitan yang mereka hadapi, dan memberikan waktu bagi guru untuk membantu siswa. Karena kode program yang diberikan pendek, guru dapat menyiapkan kode program tambahan untuk diberikan kepada siswa yang telah berhasil menyelesaikan dalam waktu singkat, atau memancing siswa untuk memodifikasi kode program yang diberikan.

Pemanasan

Guru menjalankan program yang akan dibuat.

Kebutuhan Sarana dan Prasarana

Aktivitas ini memerlukan komputer.

Kegiatan Inti

1. (5 Menit) Guru membuka kelas dan dapat melakukan pemanasan.
2. (5 Menit) Guru mendistribusikan berkas instalasi kepada siswa.
3. (20 Menit) Guru mengarahkan siswa ke aktivitas **AP-K10-04-U**: Membuat Program Pertama dengan Bahasa C.
4. (45 Menit) Guru meminta siswa mengetikkan kode program yang ada pada buku, dan menjalankannya.

5. (10 Menit) Guru mengulas poin-poin yang membuat siswa kesulitan.
6. (5 Menit) Guru menutup kelas dan mengarahkan siswa pada aktivitas refleksi.

Aspek Kreativitas

Siapkan kode program tambahan, atau tantangan tambahan yang dapat diberikan pada siswa yang telah selesai melaksanakan aktivitas dengan waktu singkat.

5. Pertemuan 5: Mengenal Algoritma dan Pemrograman (3 JP)

Tujuan Pembelajaran:

Siswa memahami konsep input-output dan mampu menuliskan program sederhana yang membaca dan menulis**.**

Apersepsi

Agar dapat digunakan oleh manusia, program harus bisa berkomunikasi, salah satunya dengan membaca masukan (*input*) dan menulis keluaran (*output*). Akan tetapi, program yang dibuat belum bisa memahami bahasa natural manusia sehingga masukan dan keluaran perlu disertai dengan 'cara membaca' dan 'cara menulis'. Hal ini dimungkinkan dengan adanya spesifikasi format. Aktivitas ini diberikan untuk memberikan pemahaman bagi siswa mengenai penggunaan spesifikasi format tersebut.

Pemanasan

Guru disarankan untuk mendemokan program komputer yang dapat membaca dan menulis berdasarkan nilai yang telah dibaca.

Kebutuhan Sarana dan Prasarana

Aktivitas ini memerlukan komputer.

Kegiatan Inti

1. (5 Menit) Guru membuka kelas dan dapat melakukan pemanasan.
2. (15 Menit) Guru menyampaikan materi dan dapat memberikan demonstrasi langsung mengenai program yang akan dibuat pada latihan ini.
3. (70 Menit) Guru mengarahkan siswa ke aktivitas Ayo, Kita Berlatih 3: Menulis dan Memperbaiki Program. Biarkanlah siswa mengerjakan

latihan mandiri. Guru memberikan umpan balik dan *scaffolding* pada siswa yang mengalami kesulitan.

4. (15 Menit) Minta siswa secara berpasangan mengecek kode dan program yang telah dibuat oleh siswa lainnya. Mintalah siswa menguji program rekan mereka dengan menggunakan kasus uji yang diberikan, atau dibuat sendiri oleh siswa. Setelah itu, pancing siswa untuk memperbaiki program apabila terdapat kesalahan pada program.
5. (15 Menit) Guru dapat membahas aktivitas dengan meminta beberapa siswa menjelaskan hasil pekerjaannya. Hal ini akan melatih siswa dalam mengomunikasikan suatu artefak pemrograman dengan rekannya.
6. (10 Menit) Guru dapat mengulas kembali beberapa kesalahan umum yang terjadi pada saat latihan pemrograman. Setelah itu, siswa dapat diminta untuk memperbaiki program di sisa waktu, atau dikerjakan kembali ke rumah.
7. (5 Menit) Guru menutup kelas dan mengarahkan siswa pada aktivitas refleksi. Kode hasil pekerjaan dapat dikumpulkan untuk diperiksa.

Pembahasan Ayo, Berlatih 3

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

Penjelasan singkat:

1. Pada soal, siswa diperintahkan untuk memasukkan tiga buah bilangan. Bilangan pertama ialah bilangan bulat (integer), bilangan kedua ialah bilangan desimal (float) dan bilangan ketiga ialah bilangan bulat (integer).
2. Untuk menyelesaikan soal, siswa perlu definisikan tiga buah variabel.

   int a : bilangan 1
   float b : bilangan 2
   int c : bilangan 3
3. Baca ketiga bilangan tersebut menggunakan **`scanf("%d %f %d", &a, &b, &c);`**,
4. Hasil masukan akan dicetak sesuai format yang diperintahkan. Untuk mendapat baris baru setelah satu variabel tercetak, gunakan **`'\n'`**.

5. Untuk mendapat dua bilangan di belakang koma, gunakan format **`%.2f`** saat ingin mencetak bilangan desimal.

Contoh Solusi Program :

```
/*
 * Program berlatih 1: mencetak tiga bilangan
 */
#include <stdio.h>
int main() {
      int a, c;
      float b;
      scanf("%d %f %d", &a, &b, &c);
      printf("%d\n%.2f\n%d\n", a,b,c);
      return 0;
}
```

Contoh Kasus Uji:

No.	Masukan	Keluaran	Keterangan
1	10 20.1235 30	10 20.12 30	contoh kasus yang ada di soal
2	23 56 50	23 56.00 50	contoh kasus uji yang ada di tengah jangkauan batasan input

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

Penjelasan singkat:

1. Kode Intan memiliki kesalahan pada penggunaan tanda kutip di tengah kalimat. “Andi berkata, “Satu, dua, tiga!”.
2. Untuk memperbaiki kode tersebut, siswa dapat menambahkan format untuk menggunakan kutip di tengah kalimat yaitu ‘\’.

Contoh Solusi Program:

```
/*
 * Program berlatih 2: memperbaiki kode Intan
 */
#include <stdio.h>

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

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

Penjelasan singkat:

1. Siswa perlu mendeklarasikan variabel terlebih dahulu sebelum membaca masukan, dengan menambah baris 3.
2. Menambahkan & di depan variabel a dan b pada baris 4 saat membaca masukan dari pengguna.
3. Ubah **\d** pada kode yang salah menjadi **\n** untuk membuat baris baru setiap kalimat seperti pada baris 5 dan 6.

Contoh Solusi Program :

```
/*
 * Program berlatih 3: memperbaiki kode program
 */

1 #include <stdio.h>
2 int main() {
3     char a, b;
4     scanf("%c %c", &a, &b);
5     printf("Bilangan pertama: %c\n", a);
6     printf("Bilangan kedua: %c\n", b);
7     return 0;
8 }
```

6. Pertemuan 6: Ekspresi (3 JP)

Tujuan Pembelajaran:

Siswa memahami konsep variabel dan ekspresi dan mengaplikasikannya dalam bentuk program.

Apersepsi

Program memproses data yang diberikan menggunakan suatu ekspresi. Pada bagian ini, siswa diperkenalkan pada ekspresi-ekspresi yang dapat berjalan di dalam suatu program. Sebagian besar ekspresi ini telah dikenal oleh siswa, terutama di mata pelajaran Matematika. Akan tetapi, ada beberapa perbedaan sintaks yang perlu diketahui oleh siswa.

Pemanasan

Guru disarankan untuk mendemokan program komputer yang dapat melakukan perhitungan-perhitungan sederhana, berdasarkan masukan yang diberikan oleh pengguna.

Kebutuhan Sarana dan Prasarana

Aktivitas ini memerlukan komputer.

Kegiatan Inti

1. (5 Menit) Guru membuka kelas dan dapat melakukan pemanasan.
2. (15 Menit) Guru menyampaikan materi dan dapat memberikan demonstrasi langsung mengenai program yang akan dibuat pada latihan ini.
3. (70 Menit) Guru mengarahkan siswa ke aktivitas Ayo, Kita Berlatih 4: Latihan Ekspresi. Biarkanlah siswa mengerjakan latihan mandiri. Guru memberikan umpan balik dan *scaffolding* pada siswa yang mengalami kesulitan.
4. (15 Menit) Minta siswa secara berpasangan mengecek kode dan program yang telah dibuat oleh siswa lainnya. Mintalah siswa menguji program rekan mereka dengan menggunakan kasus uji yang diberikan, atau dibuat sendiri oleh siswa. Setelah itu, pancing siswa untuk memperbaiki program apabila terdapat kesalahan pada program.

5. (15 Menit) Guru dapat membahas aktivitas dengan meminta beberapa siswa menjelaskan hasil pekerjaannya. Hal ini akan melatih siswa dalam mengomunikasikan suatu artefak pemrograman dengan rekannya.
6. (10 Menit) Guru dapat mengulas kembali beberapa kesalahan umum yang terjadi pada saat latihan pemrograman. Setelah itu, siswa dapat diminta untuk memperbaiki program di sisa waktu, atau dikerjakan kembali ke rumah.
7. (5 Menit) Guru menutup kelas dan mengarahkan siswa pada aktivitas refleksi. Kode hasil pekerjaan dapat dikumpulkan untuk diperiksa.

Pembahasan Ayo Berlatih 4

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

Penjelasan singkat:

1. Pada soal ini, siswa diminta untuk menghitung luas tanah berbentuk segitiga siku-siku dengan 2 bilangan masukan. Bilangan pertama merupakan panjang alas dan bilangan kedua merupakan tinggi.
2. Untuk menghitung luas tanah tersebut, perlu menggunakan rumus luas segitiga berupa alas x tinggi / 2.
3. Hasil keluaran berupa luas tanah dengan 2 angka di belakang koma. Gunakan format **%.2f** untuk mencetak bilangan desimal dengan 2 angka di belakang koma.

Contoh Solusi Program:

```
/*
 * Program berlatih 4.1: menghitung luas tanah
 */
#include <stdio.h>
int main() {
      float alas, tinggi, luas;
      scanf("%f %f", &alas, &tinggi);
      luas = (alas * tinggi) / 2;
      printf("%.2f\n", luas);
      return 0;
}
```

Contoh Kasus Uji:

No.	Masukan	Keluaran	Keterangan
1	20 40	400.00	Contoh kasus yang ada di soal
2	25 35	437.50	Contoh kasus yang hasil luasnya bilangan desimal

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

Penjelasan singkat:

1. Siswa diminta untuk menghitung luas persegi sesuai dengan diagram alir 1.
2. Masukan berupa sebuah bilangan yang disimpan pada variabel sisi.
3. Luas persegi dihitung dengan mengalikan bilangan sisi dengan bilangan sisi itu sendiri.
4. Hasil perkalian dicetak dengan perintah **`printf`**.

Contoh Solusi Program:

```
/*
 * Program berlatih 4.2: menghitung luas persegi
 */
#include <stdio.h>
int main() {
      int sisi, luas;
      scanf("%d", &sisi);
      luas = sisi * sisi;
      printf("%d\n", luas);
      return 0;
}
```

Contoh Kasus Uji:

No.	Masukan	Keluaran	Keterangan
1	100	10000	Contoh kasus uji
2	25	625	Contoh kasus uji

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

Penjelasan singkat:

1. Siswa diminta untuk mendeklarasikan variabel a, b, c, d terlebih dahulu.
2. Variabel a dan b untuk bilangan masukan, variabel c untuk hasil pembagian, dan variabel d untuk sisa pembagian (modulo).
3. Menghitung hasil pembagian dilakukan dengan cara bilangan a dibagi bilangan b.
4. Menghitung sisa pembagian dilakukan dengan cara bilangan a modulo bilangan b.
5. Hasil pembagian dan sisa pembagian kemudian dicetak dengan memberikan **\n** pada tiap variabel **c** dan **d** agar berpindah ke baris baru.

Contoh Solusi Program:

```
/*
 * Program berlatih 4.3: hasil bagi dan sisa pembagian
 */
#include <stdio.h>
int main() {
      long long int a, b, c, d;
      scanf("%lld %lld", &a, &b);
      c = a / b;
      d = a % b;
      printf("%lld\n %lld\n ", c, d);
      return 0;
}
```

Contoh Kasus Uji:

No.	Masukan	Keluaran	Keterangan
1	1000 3	333 1	Contoh kasus yang ada di soal
2	9 13	0 9	Contoh kasus jika bilangan pembagi lebih besar
3	4000 4	1000 0	Contoh kasus jika tidak ada sisa pembagian

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

Penjelasan singkat:

1. Tanda "<", ">", ">=", "==", "!=" secara berurutan merupakan operator "kurang dari", "lebih dari", "lebih dari sama dengan", "sama dengan", "tidak sama dengan". Tanda "||" merupakan operator "atau", tanda "&&" merupakan operator "dan".
2. Operator "atau" bernilai benar jika sedikitnya satu dari kedua pernyataan yang ada bernilai benar. Operator "dan" bernilai benar jika kedua pernyataan bernilai benar.

Solusi:

```
/*
 * Solusi berlatih 4: untuk benar atau salah?
 */
1. Benar
2. Benar
3. Salah
4. Benar
```

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

Penjelasan singkat:

1. Kesalahan pertama terdapat pada baris include, dimana seharusnya stdio.h bukan studio.h.
2. Kode program tersebut dapat dipersingkat pada bagian deklarasi variabel dengan menggunakan koma sajakarena tipe data dari ketiga variabel tersebut sama (**`float`**).
3. Pada baris scanf, terdapat kesalahan pada "**`jr2`**", dimana seharusnya ada "**`&`**" sebelum **`jr2`**.
4. Pada baris printf, tambahkan "**`\n`**" setelah "**`%.2f`**" pertama untuk berpindah ke baris baru.

5. Kode program tersebut ialah kode program untuk menghitung luas pada variabel "**l**" dan keliling lingkaran pada variabel "**o**" dengan bilangan masukan berupa jari-jari.

Contoh Solusi Program:

```
/*
 * Program berlatih 5: percantik kode program ini!
 */
#include <stdio.h>
int main() {
      float jr2, l, O;
      scanf("%f", &jr2);
      l = 3.14*jr2*jr2;
      O = 2*3.14*jr2;
      printf("%.2f\n %.2f\n",l, O);
      return 0;
}
```

Contoh Kasus Uji:

No.	Masukan	Keluaran	Keterangan
1	10	314.00 62.80	Contoh kasus uji
2	25	1962.50 157.00	Contoh kasus uji
3	12.5	490.62 78.50	Contoh kasus uji

7. 7. Pertemuan 7: Struktur Kontrol Keputusan (3 JP)

Tujuan Pembelajaran:

Siswa memahami konsep struktur kontrol keputusan dan mengaplikasikannya dalam bentuk program.

Apersepsi

Program dapat memiliki perilaku yang berbeda bergantung pada masukan yang diterimanya. Misalnya, suatu pembagian dengan nilai 0 tidak dieksekusi

oleh program karena membuat program berhenti secara tidak wajar. Pada saat login ke akun digital, apabila nama pengguna dan kata sandi salah, aplikasi tidak akan memberikan akses ke dalamnya. Hal ini dapat dimungkinkan dengan adanya struktur kontrol keputusan.

Pemanasan

Guru disarankan untuk mendemokan program komputer yang menggunakan struktur kontrol keputusan.

Kebutuhan Sarana dan Prasarana

Aktivitas ini memerlukan komputer.

Kegiatan Inti

1. (5 Menit) Guru membuka kelas dan dapat melakukan pemanasan.
2. (15 Menit) Guru menyampaikan materi dan dapat memberikan demonstrasi langsung mengenai program yang akan dibuat pada latihan ini.
3. (70 Menit) Guru mengarahkan siswa ke aktivitas Ayo, Kita Berlatih 5: Latihan Struktur Kontrol Keputusan. Biarkanlah siswa mengerjakan latihan mandiri. Guru memberikan umpan balik dan *scaffolding* pada siswa yang mengalami kesulitan.
4. (15 Menit) Minta siswa secara berpasangan mengecek kode dan program yang telah dibuat oleh siswa lainnya. Mintalah siswa menguji program rekan mereka dengan menggunakan kasus uji yang diberikan, atau dibuat sendiri oleh siswa. Setelah itu, pancing siswa untuk memperbaiki program apabila terdapat kesalahan pada program.
5. (15 Menit) Guru dapat membahas aktivitas dengan meminta beberapa siswa menjelaskan hasil pekerjaannya. Hal ini akan melatih siswa dalam mengomunikasikan suatu artefak pemrograman dengan rekannya.
6. (10 Menit) Guru dapat mengulas kembali beberapa kesalahan umum yang terjadi pada saat latihan pemrograman. Setelah itu, siswa dapat diminta untuk memperbaiki program di sisa waktu, atau dikerjakan kembali ke rumah.

7. (5 Menit) Guru menutup kelas dan mengarahkan siswa pada aktivitas refleksi. Kode hasil pekerjaan dapat dikumpulkan untuk diperiksa.

Pembahasan Ayo, Berlatih 5

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

Penjelasan singkat :

1. Siswa dapat melihat kembali diagram alir 3 pada Buku Siswa.
2. Deklarasikan variabel pembilang, penyebut, dan hasil. Variabel hasil digunakan untuk menampung hasil pembagian dari pembilang dan penyebut.
3. Buat kondisi if untuk mengecek penyebut = 0. Jika kondisi terpenuhi, cetak kalimat "Penyebut tidak boleh nol.". Jika kondisi tidak terpenuhi, cetak hasil pembagian dari pembilang dan penyebut. Cetak hasil dengan format dua bilangan di belakang koma.

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

```
/*
 * Program berlatih 1: Membuat program membagi bilangan
 */
#include <stdio.h>
int main() {
      float pembilang, penyebut, hasil;
      scanf("%f %f", &pembilang, &penyebut);

      if(penyebut == 0){
            printf("Penyebut tidak boleh nol.\n");
      }
      else {
            hasil = pembilang / penyebut;
            printf("%.2f\n", hasil);
      }
      return 0;
}
```

Contoh Kasus Uji:

No.	Masukan	Keluaran	Keterangan
1	12 3	4.00	Contoh kasus jika dapat membagi
2	12 5	2.40	Contoh kasus jika dapat membagi
3	4 0	Penyebut tidak boleh nol.	Contoh kasus jika penyebut nol

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

Penjelasan singkat :

1. Siswa perlu mendeklarasikan variabel n terlebih dahulu.
2. Masukkan bilangan n yang sesuai deskripsi soal dengan scanf.
3. Lakukan pengecekan jika nilai n lebih dari 0 (1,2,3.. dst),kondisi terpenuhi dan cetak kalimat Bilangan Bulat Positif.

Contoh Solusi Program:

```
#include <stdio.h>
/*
 * Program berlatih: Apakah Bilangan bulat positif
 */
int main() {
      int n;
      scanf("%d", &n);
      if(n > 0){
            printf("Bilangan Bulat Positif\n");
      }
      return 0;
}
```

Contoh Kasus Uji:

No.	Masukan	Keluaran	Keterangan
1	3	Bilangan Bulat Positif	Contoh kasus pada soal
2	-10	<Tidak ada>	Contoh kasus pada soal
3	0	<Tidak ada>	Contoh kasus jika n adalah nol
4	10	Bilangan Bulat Positif	Contoh kasus n lebih dari 0

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

Penjelasan singkat :

1. Siswa mendeklarasikan variabel n untuk bilangan bulat (integer). Kemudian, membuat masukkan berupa bilangan bulat.
2. Lakukan pengecekan kondisi sebanyak tiga kali, yaitu:
   a. Saat n > 0 : Bilangan bulat positif
   b. Saat n < 0 : Bilangan bulat negatif
   c. Saat kedua kondisi diatas tidak terpenuhi akan mencetak bilangan bulat nol

Contoh Solusi Program :

```
/*
 * Program berlatih : apakah positif, negatif, nol
 */
#include <stdio.h>
int main() {
      int n;
      scanf("%d", &n);
      if(n > 0){
            printf("Bilangan Bulat Positif\n");
      }
      else if(n < 0){
            printf("Bilangan Bulat Negatif\n");
      }
```

```
        else {
                printf("Bilangan Bulat Nol\n");
        }
        return 0;
}
```

Contoh Kasus Uji:

No.	Masukan	Keluaran	Keterangan
1	3	Bilangan Bulat Positif	Contoh kasus pada soal
2	-5	Bilangan Bulat Negatif	Contoh kasus pada soal
3	0	Bilangan Bulat Nol	Contoh kasus n = 0
4	-101	Bilangan Bulat Negatif	Contoh kasus kurang dari 0

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

Penjelasan singkat:

1. Siswa dapat menggunakan struktur **`switch-case`** karena kondisi memiliki banyak cabang.
2. Siswa mendeklarasikan variabel bilangan bulat n. Kemudian, membuat masukan untuk **`n`**.
3. Siswa membuat kondisi pengecekan nilai n yang sesuai dengan nomor bulan dari Januari sampai Desember menggunakan struktur **`switch-case`**. Jika nilai n melebihi atau kurang dari nilai 1 - 12, program mencetak "Tidak ada bulan yang sesuai".

Contoh solusi program :

```
/*
 * Program berlatih : Nama Bulan
 */
#include <stdio.h>
int main() {
      int n;
```

```
scanf("%d", &n);
switch(n){
      case 1:
            printf("Januari\n");
            break;
      case 2:
            printf("Februari\n");
            break;
      case 3:
            printf("Maret\n");
            break;
      case 4:
            printf("April\n");
            break;
      case 5:
            printf("Mei\n");
            break;
      case 6:
            printf("Juni\n");
            break;
      case 7:
            printf("Juli\n");
            break;
      case 8:
            printf("Agustus\n");
            break;
      case 9:
            printf("September\n");
            break;
      case 10:
            printf("Oktober\n");
            break;
```

```
            case 11:
                  printf("November\n");
                  break;
            case 12:
                  printf("Desember\n");
                  break;
            default:
                  printf("Tidak ada bulan yang sesuai.\n");
                  break;
      }
      return 0;
}
```

Contoh Kasus Uji:

No.	Masukan	Keluaran	Keterangan
1	1	Januari	Contoh kasus pada soal
2	3	Maret	Contoh kasus pada soal
3	13	Tidak ada bulan yang sesuai.	Contoh kasus pada soal
4	0	Tidak ada bulan yang sesuai.	Contoh kasus nilai n = 0

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

Penjelasan singkat:

1. Siswa mengikuti diagram alir yang sudah disediakan pada soal.
2. Langkah awal ialah membaca masukan dari pengguna dengan scanf.
3. Kemudian, lakukan pengecekan terhadap ketiga variabel sisi dan memasukkan nilainya ke variabel terbesar.
4. Setelah program mendapat nilai terbesar, hitung jumlah dari ketiga sisi dan simpan pada variabel total.
5. Lakukan pengecekan kedua untuk mengecek sisi terpanjang kurang dari jumlah kedua sisi lainnya. Mendapat jumlah kedua sisi lainnya ialah

dengan mengurangi total seluruh sisi dengan nilai sisi terbesar (`total - terbesar`).

Contoh Solusi Program:

```
/*
 * Program berlatih : Mengecek sisi segitiga
 */
#include <stdio.h>
int main() {

      int a,b,c;
      int terbesar, total;

      scanf("%d %d %d", &a, &b, &c); //Membaca masukkan

      if(a >= b && a >= c){ //Mengecek sisi terbesar
            terbesar = a;
      }
      else if(b >= a && b >= c){
            terbesar = b;
      }
      else {
            terbesar = c;
      }

      total = a + b + c; //Menghitung total seluruh sisi

      if(terbesar < total - terbesar){ //Mengecek sisi
terpanjang
            printf("valid\n");
      }
      else{
            printf("tidak valid\n");
      }

      return 0;
}
```

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

Penjelasan Singkat:

1. Siswa diperintahkan membuat uji kasus untuk program yang sudah dibuat pada program 5 diatas.
2. Uji kasus yang dilakukan dengan membuat beberapa nilai masukan yang akan menelusuri setiap kondisi yang ada.
3. Cobalah membuat beberapa masukan yang akan menghasilkan tidak valid, valid, nilai terbesar c, nilai terbesar b dan nilai terbesar a.

Contoh Kasus Uji:

No.	Masukan	Keluaran	Keterangan
1	1 5 8	tidak valid	contoh kasus c yang terbesar
2	8 12 5	valid	contoh kasus b yang terbesar
3	-2 -7 -6	tidak valid	Contoh kasus nilai negatif
4	13 8 10	valid	Contoh kasus a yang terbesar

8. Pertemuan 8: Struktur Kontrol Perulangan (3 JP)

Tujuan Pembelajaran:

Siswa memahami konsep struktur kontrol perulangan dan mengaplikasikannya dalam bentuk program.

Apersepsi

Komputer memiliki kemampuan untuk melakukan suatu hal secara berulang-ulang dengan konsisten. Hal ini dimungkinkan dengan adanya struktur kontrol perulangan. Ada beberapa variasi dalam merancang suatu perulangan pasti berhenti. Pada aktivitas ini, siswa akan diperkenalkan pada beberapa penggunaan umum dari kontrol perulangan pada program.

Pemanasan

Guru disarankan untuk mendemokan program komputer yang dapat melakukan suatu operasi secara berulang-ulang. Misalnya, mencetak barisan matematika.

Kebutuhan Sarana dan Prasarana

Aktivitas ini memerlukan komputer.

Kegiatan Inti

1. (5 Menit) Guru membuka kelas dan dapat melakukan pemanasan.
2. (15 Menit) Guru menyampaikan materi dan dapat memberikan demonstrasi langsung mengenai program yang akan dibuat pada latihan ini.
3. (70 Menit) Guru mengarahkan siswa ke aktivitas Ayo, Kita Berlatih 6: Latihan Struktur Kontrol Perulangan. Biarkanlah siswa mengerjakan latihan mandiri. Guru memberikan umpan balik dan *scaffolding* pada siswa yang mengalami kesulitan.
4. (15 Menit) Minta siswa secara berpasangan mengecek kode dan program yang telah dibuat oleh siswa lainnya. Mintalah siswa menguji program rekan mereka dengan menggunakan kasus uji yang diberikan, atau dibuat sendiri oleh siswa. Setelah itu, pancing siswa untuk memperbaiki program apabila terdapat kesalahan pada program.
5. (15 Menit) Guru dapat membahas aktivitas dengan meminta beberapa siswa menjelaskan hasil pekerjaannya. Hal ini akan melatih siswa dalam mengomunikasikan suatu artefak pemrograman dengan rekannya.
6. (10 Menit) Guru dapat mengulas kembali beberapa kesalahan umum yang terjadi pada saat latihan pemrograman. Setelah itu, siswa dapat diminta untuk memperbaiki program di sisa waktu, atau dikerjakan kembali ke rumah.
7. (5 Menit) Guru menutup kelas dan mengarahkan siswa pada aktivitas refleksi. Kode hasil pekerjaan dapat dikumpulkan untuk diperiksa.

Pembahasan Ayo, Berlatih 6

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

Penjelasan Singkat:

1. Siswa dapat melihat kembali diagram alir 4 pada Buku Siswa.
2. Tentukan nilai n.
3. Selama n bukan 0, nilai n akan terus dicetak oleh program.

Kode Program:

```
/*
 * Program berlatih 6: Menghitung mundur
 */
#include <stdio.h>
int main() {
      int n;

      scanf("%d", &n); // membaca inputan

      while (n!=0){
      printf("%d", n);  // mencetak nilai saat ini
      if (n!= 1) printf(" ");
      n=n-1;}  // mengurangi nilai saat ini

      return 0;
}
```

Contoh Kasus Uji:

No.	Masukan	Keluaran	Keterangan
1	5	5 4 3 2 1	Contoh kasus pada soal
2	7	7 6 5 4 3 2 1	Contoh kasus jika n ialah 1
3	1	1	Contoh kasus jika n ialah 5
4	2	2 1	Contoh kasus jika hasil dibelakang koma bukan .00

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

Penjelasan Singkat:

1. n dan bilangan inputan merupakan bilangan bulat.
2. Hasil akhir merupakan bilangan riil dengan dua angka dibelakang titik decimal.
3. Untuk menyimpan nilai dari n total inputan, diperlukan 1 variabel bantuan.

4. Dikarenakan bilangan bulat belum tentu dapat dibagi habissehingga apabila jenis hasil akhir masih bertipe *int,* variabel penyimpan hasil harus dalam bentuk *float/double* agar sesuai dengan poin 2

Kode Program:

```
/*
 * Program berlatih  Menghitung rataan
 */
#include <stdio.h>
int main() {
      int n,i,a;
      double avr,sum;
      sum =0;

      scanf("%d", &n); // banyaknya input

      for (i=0;i<n;i++){
        scanf("%lf", &a); //membaca nilai mencari rataan
        sum = sum + a; //menyimpan nilai dalam variabel
      }
      avr = sum/n;   //membagi total dengan n(mencari
rataan)

      printf("%.2lf\n ", avr);

return 0;
}
```

Contoh Kasus Uji:

No.	Masukan	Keluaran	Keterangan
1	3 10 20 30	20.00	Contoh kasus pada soal
2	1 20	20.00	Contoh kasus jika n ialah 1
3	5 5 10 15 20 25	15.00	Contoh kasus jika n ialah 5
4	4 1 4 6 8	4.75	Contoh kasus jika hasil dibelakang koma bukan .00

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

Penjelasan Singkat:

1. n adalah total bilangan yang akan dimasukkan.
2. Untuk menentukan bilangan terbesar,perlu sebuah variabel untuk menyimpannya, yang diatur dengan nilai 0.
3. Apabila bilangan terbesar lebih kecil dari nilai inputan, variabel bilangan terbesar digantikan dengan bilangan tersebut.
4. Proses ini akan terus berlangsung hingga proses iterasi selesai.

Kode Program:

```
/*
 * Program berlatih Mencari Bilangan Terbesar
 */
#include <stdio.h>
int main() {
      int n,a,terbesar;

      terbesar =0;

      scanf("%d", &n);  // banyaknya input

      while (n!= 0) {
         scanf("%d", &a); // memasukkan nilai bilangan
         if(terbesar<a)terbesar = a; // mengecek kondisi
                                        terbesar
      }

      printf("%.d\n ", terbesar);

      return 0;
}
```

Contoh Kasus Uji:

No.	Masukan	Keluaran	Keterangan
1	4 1 3 2 4	4	Contoh kasus pada soal
2	1 20	20	Contoh kasus jika n ialah 1
3	10 3 4 2 5 7 18 1 6 7 17	18	Contoh kasus jika n ialah 10
4	7 5 2 7 8 10 2 10	4.75	Contoh kasus jika hasil bilangan terbesar ada 2

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

Penjelasan Singkat:

1. Deklarasikan lebih dulu 3 variabel bernama kecil, sedang, dan besar yang akan dipakai untuk menyimpan jumlah kembang kol berukuran kecil, sedang, dan besar. Di saat awal, ketiga variabel tersebut diinisialisasi (diisi) dengan nilai 0.
2. Deklarasikan juga variable berat yang akan dipakai untuk menyimpan berat kembang kol yang sedang disortir.
3. Data berat kembang kol dibaca satu persatu. Setiap kali dibaca, lalu dicek apakah beratnya termasuk kategori kecil, sedang, atau besar.
4. Jika berat kembang kol termasuk kategori kecil,isi variable kecil ditambah 1; jika beratnya sedang,isi variable sedang ditambah 1; dan jika beratnya termasuk kategori besar,isi variable besar ditambah 1.
5. Proses nomor 3 dan 4 diulangi selama berat bunga kol tidak sama dengan -1.

Kode Program:

```
/*
 * Program berlatih: Mesin Sortir Kembang Kol
 */
#include <stdio.h>
int main() {
      int kecil,sedang,besar;
```

```
      float berat;
      kecil=0;
      sedang=0;
      besar=0;

      scanf("%f", &berat);  // baca data berat bunga
kol pertama

      while (berat!=-1) { //selama berat tidak sama
                  //dengan -1 ulangi proses di bawah ini
            if(berat<50) kecil++; //menambah jumlah kecil
                 else if(berat<200 && berat>50) sedang++;
                              //menambah jumlah sedang
                       else besar++; // menambah jumlah
                                          besar
            scanf("%f", &berat);  // baca data berat bunga
                                       kol berikutnya
      }

      printf("%.d\n%.d\n%.d\n", kecil,sedang,besar);

return 0;
}
```

Contoh Kasus Uji:

No.	Masukan	Keluaran	Keterangan
1	100.0 20.5 300.1 40.1 -1	2 1 1	Contoh kasus dengan 4 data
2	30.0 50.5 250.2 -1	1 1 1	Contoh kasus dengan 3 data
3	60.0 20.5 70.5 40.1 -1	2 2 0	Contoh kasus dengan 4 data, salah satu kategori jumlahnya 0

4	70.5 60.5 70.0 140.0 -1	0 4 0	Contoh kasus dengan 4 data, dengan kategori yang sama

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

Penjelasan Singkat:

1. n ialah total bilangan yang akan dimasukkan sebagai bentuk X.
2. Pola pada contoh kasus ialah memperlihatkan posisi bintang akan berada pada bagian yang ujung, yang kemudian akan berpindah 1 tempat ke sebelahnya.
3. Apabila inputan merupakan bilangan ganjil, bagian tengah pola akan memiliki tepat 1 bintang di tengah. Apabila bilangan genap,bagian tengahnya akan memiliki 2 bintang.
4. Bagian bawah dari pola ialah cerminan terbalik dari bagian atas pola.

Kode Program:

```
/*
 * Program berlatih : Memperbaiki program
 */
#include <stdio.h>
int main() {
      int n, i, j;
      scanf("%d", &n);

      for(i=0; i>n; i++){   //membentuk matriks i
            for(j=0; j<n; j++){    //membentuk matriks j
             if(j==i || j==n-i-1)printf("*"); //mencetak
                                    * jika memenuhi syarat
             else printf("-"); //mencetak -
            }
            printf("\n");
      }
}
```

Contoh Kasus Uji:

No.	Masukan	Keluaran	Keterangan
1	5	*---* _*_*_ __*__ _*_*_ *---*	Contoh kasus pada soal
2	6	*----* _*__*_ __**__ __**__ _*__*_ *----*	Contoh kasus pada soal
3	20	*------------------* _*----------------*_ __*--------------*__ ___*------------*___ ____*----------*____ _____*--------*_____ ______*------*______ _______*----*_______ _______*__*_______ ________**________ ________**________ _______*__*_______ _______*----*_______ ______*------*______ _____*--------*_____ ____*----------*____ ___*------------*___ __*--------------*__ _*----------------*_ *------------------*	Contoh kasus jika n adalah 20
3	5	*---* _*_*_ __*__ _*_*_ *---*	Contoh kasus pada soal
4	4	*--* _**_ _**_ *--*	Contoh kasus jika n adalah 4

9. Pertemuan 9: Fungsi (3 JP)

Tujuan Pembelajaran:

Siswa memahami implementasi fungsi dalam program.

Apersepsi

Fungsi menjadi penting sebagai cara menerapkan abstraksi dalam membuat sebuah program. Dengan fungsi, kode program yang kompleks dapat dibuat lebih mudah dibaca dengan mengumpulkan kode-kode yang memiliki tujuan tertentu ke dalam suatu fungsi. Oleh karena itu, aktivitas ini dirancang untuk mengenalkan siswa pada penggunaan fungsi.

Pemanasan

Guru disarankan untuk membandingkan dua buah program komputer yang menggunakan fungsi dan yang tidak. Keduanya memiliki tujuan yang sama.

Kebutuhan Sarana dan Prasarana

Aktivitas ini memerlukan komputer.

Kegiatan Inti

1. (5 Menit) Guru membuka kelas dan dapat melakukan pemanasan.
2. (15 Menit) Guru menyampaikan materi dan dapat memberikan demonstrasi langsung mengenai program yang akan dibuat pada latihan ini.
3. (70 Menit) Guru mengarahkan siswa ke aktivitas Ayo, Kita Berlatih 7: Latihan Fungsi. Biarkanlah siswa mengerjakan latihan mandiri. Guru memberikan umpan balik dan *scaffolding* pada siswa yang mengalami kesulitan.
4. (15 Menit) Minta siswa secara berpasangan mengecek kode dan program yang telah dibuat oleh siswa lainnya. Mintalah siswa menguji program rekan mereka dengan menggunakan kasus uji yang diberikan, atau dibuat sendiri oleh siswa. Setelah itu, pancing siswa untuk memperbaiki program apabila terdapat kesalahan pada program.
5. (15 Menit) Guru dapat membahas aktivitas dengan meminta beberapa siswa menjelaskan hasil pekerjaannya. Hal ini akan melatih siswa dalam mengomunikasikan suatu artefak pemrograman dengan rekannya.

6. (10 Menit) Guru dapat mengulas kembali beberapa kesalahan umum yang terjadi pada saat latihan pemrograman. Setelah itu, siswa dapat diminta untuk memperbaiki program di sisa waktu, atau dikerjakan kembali ke rumah.
7. (5 Menit) Guru menutup kelas dan mengarahkan siswa pada aktivitas refleksi. Kode hasil pekerjaan dapat dikumpulkan untuk diperiksa.

Pembahasan Ayo, Berlatih 7

Soal 1. Buatlah kode program dari diagram alir dua pada bagian algoritma, yaitu menghitung luas permukaan kubus.

Penjelasan Singkat:

1. Siswa diminta untuk menghitung luas permukaan kubus sesuai dengan diagram alir 2.
2. Masukan berupa sebuah bilangan yang disimpan pada variabel sisi.
3. Fungsi luas persegi dihitung dengan mengalikan bilangan sisi dengan bilangan sisi itu sendiri.
4. Luas permukaan kubus dihitung dengan memanggil fungsi luas persegi dan kemudian dikali dengan 6 (permukaan kubus berupa persegi sebanyak 6 buah).
5. Hasil yang didapatkan dicetak dengan perintah printf.

Contoh Solusi Program:

```
/*
 * Program  menghitung luas permukaan kubus dengan fungsi
 */
#include <stdio.h>

int luasPersegi(int sisi) {
//fungsi untuk menghitung luas persegi
      int luas;
      luas = sisi * sisi;
      return luas;
}

int main() {
      int n, luas, luasPermukaan;
      scanf("%d", &n);
      luas = luasBujurSangkar(n);
      luasPermukaan = luas * 6;
      printf("%d\n", luasPermukaan);
      return 0;
}
```

Contoh Kasus Uji:

No.	Masukan	Keluaran	Keterangan
1	100	60000	Contoh kasus uji
2	25	3750	Contoh kasus uji

Soal 2. Buatlah sebuah fungsi untuk menghitung luas dan keliling bangun datar, seperti persegi panjang, lingkaran, dan segitiga.

Penjelasan Singkat:

1. Siswa diminta untuk membuat fungsi menghitung luas dan keliling bangun datar.
2. Setiap fungsi memiliki tipe data yang berbeda seperti : **`int`** untuk persegi panjang, **`float`** untuk lingkaran dan segitiga.
3. Untuk menghitung luas dan keliling, digunakan rumus luas dan keliling bangun datar.
4. Hasil yang didapatkan akan dikembalikan dengan perintah return.

Contoh Solusi Program:

```
/*
 * Program untuk fungsi menghitung luas dan keliling
 * bangun datar
 */

//fungsi luas dan keliling persegi panjang
#include <stdio.h>

int luasPersegiPanjang(int panjang, int lebar) {
      int luas;
      luas = panjang * lebar;
      return luas;
}

int kelilingPersegiPanjang(int panjang, int lebar) {
      int keliling;
      keliling = 2 * (panjang + lebar);
      return keliling;
}
```

```
int main() {
      int a, b, luas, keliling;
      scanf("%d %d", &a, &b);
      luas = luasPersegiPanjang(a, b);
      keliling = kelilingPersegiPanjang(a, b);
      printf("%d\n %d\n", luas, keliling);
      return 0;
}

//fungsi luas dan keliling lingkaran
float luasLingkaran(float radius) {
      float luas;
      luas = 3.14 * radius * radius;
      return luas;
}

float kelilingLingkaran(float radius) {
      float keliling;
      keliling = 3.14 * radius * 2;
      return keliling;
}

float main() {
      float a, luas, keliling;
      scanf("%f", &a);
      luas = luasLingkaran(a);
      keliling = kelilingLingkaran(a);
      printf("%.2f\n %.2f\n", luas, keliling);
      return 0;
}
```

```
//fungsi luas dan keliling segitiga

float luasSegitiga(float alas, float tinggi) {
      float luas;
      luas = alas * tinggi / 2;
      return luas;
}

float kelilingSegitiga(float sisi1, float sisi2, float sisi3) {
      float keliling;
      keliling = sisi1 + sisi2 + sisi3;
      return keliling;
}

float main() {
      float a, t, s1, s2, s3, luas, keliling;
      scanf("%f %f %f %f %f", &a, &t, &s1, &s2, &s3);
      luas = luasSegitiga(a, t);
      keliling = kelilingSegitiga(s1, s2, s3);
      printf("%.2f\n %.2f\n", luas, keliling);
      return 0;
}
```

Contoh Kasus Uji:

No.	Masukan	Keluaran	Keterangan
1	10	314.00 62.80	Contoh kasus untuk fungsi luas dan keliling lingkaran
2	9 10 5 8 10	45.00 23.00	Contoh kasus untuk fungsi luas dan keliling segitiga
3	10 5	50 30	Contoh kasus untuk fungsi luas dan keliling persegi panjang

Soal 3. Buatlah sebuah fungsi untuk menghitung luas permukaan bangun ruang seperti balok, kerucut, bola, dan limas.

Penjelasan Singkat:

1. Siswa diminta untuk menghitung luas permukaan bangun ruang.
2. Setiap fungsi memiliki tipe data yang berbeda seperti: int untuk balok, float untuk kerucut, bola, dan limas.
3. Untuk menghitung luas permukaan bangun ruang, digunakan rumus luas permukaan dari setiap bangun ruang.
4. Hasil yang didapatkan akan dikembalikan dengan perintah return.

Contoh Solusi Program:

```
/*
 * Program berlatih fungsi menghitung luas permukaan
 * bangun ruang
 */

//fungsi luas permukaan balok
#include <stdio.h>

int luasBalok(int panjang, int lebar, int tinggi) {
      int luas;
      luas = 2 * ((panjang * lebar) + (panjang * tinggi)
      + (lebar * tinggi));
      return luas;
}

int main() {
      int a, b, c, luas;
      scanf("%d %d %d", &a, &b, &c);
      luas = luasBalok(a, b, c);
      printf("%d\n", luas);
      return 0;
}
```

```
//fungsi luas permukaan kerucut

float luasKerucut(float radius, float sisi) {
      float luas;
      luas = (3.14 * radius * radius) + (3.14 * radius
      * sisi);
      return luas;
}

float main() {
      float a, s, luas;
      scanf("%f %f", &a, &s);
      luas = luasKerucut(a, s);
      printf("%.2f\n", luas);
      return 0;
}

//fungsi luas permukaan bola

float luasBola(float radius) {
      float luas;
      luas = 3.14 * radius * radius * 4;
      return luas;
}

float main() {
      float r, luas;
      scanf("%f", &r);
      luas = luasBola(r);
      printf("%.2f\n", luas);
      return 0;
}
```

```
//fungsi luas permukaan limas

float luasLimasSegiEmpat(float sisi, float tinggiSisiTegak) {
      float luas;
      luas = (sisi * 4) + (4 * (sisi * tinggiSisiTegak
      / 2));
      return luas;
}

float main() {
      float s, t, luas;
      scanf("%f %f", &s, &t);
      luas = luasLimasSegiEmpat(s, t);
      printf("%.2f\n", luas);

//panggillah fungsi lainnya

      return 0;
}
```

Contoh Kasus Uji:

No.	Masukan	Keluaran	Keterangan
1	17 12 9	930	Contoh kasus untuk fungsi luas permukaan balok
2	7 10	373.66	Contoh kasus untuk fungsi luas permukaan kerucut
3	17.5	3846.50	Contoh kasus untuk fungsi luas permukaan bola
4	15 10.5	375.00	Contoh kasus untuk fungsi luas permukaan limas segiempat

10. Pertemuan 10: Latihan Pemrograman (6 JP)

Tujuan Pembelajaran:

Siswa mampu mengaplikasikan konsep-konsep pemrograman prosedural (baca tulis, variabel, ekspresi, struktur kontrol keputusan dan pengulangan, serta fungsi) dalam menyelesaikan persoalan yang lebih kompleks.

Apersepsi

Aktivitas ini lebih sulit dibandingkan pada pertemuan sebelumnya karena mengombinasikan kemampuan berpikir komputasional siswa dengan kemampuan menulis kode program (*coding*). Siswa akan diminta menyelesaikan yang lebih kompleks, dan perlu menggunakan proses abstraksi, dekomposisi, pengenalan pola, dan algoritma untuk menghasilkan solusi dari permasalahan yang diberikan. Kegiatan dapat dilakukan secara berpasangan.

Pemanasan

Guru disarankan untuk mengulas kembali peran program untuk *problem solving*.

Kebutuhan Sarana dan Prasarana

Aktivitas ini memerlukan komputer.

Kegiatan Inti

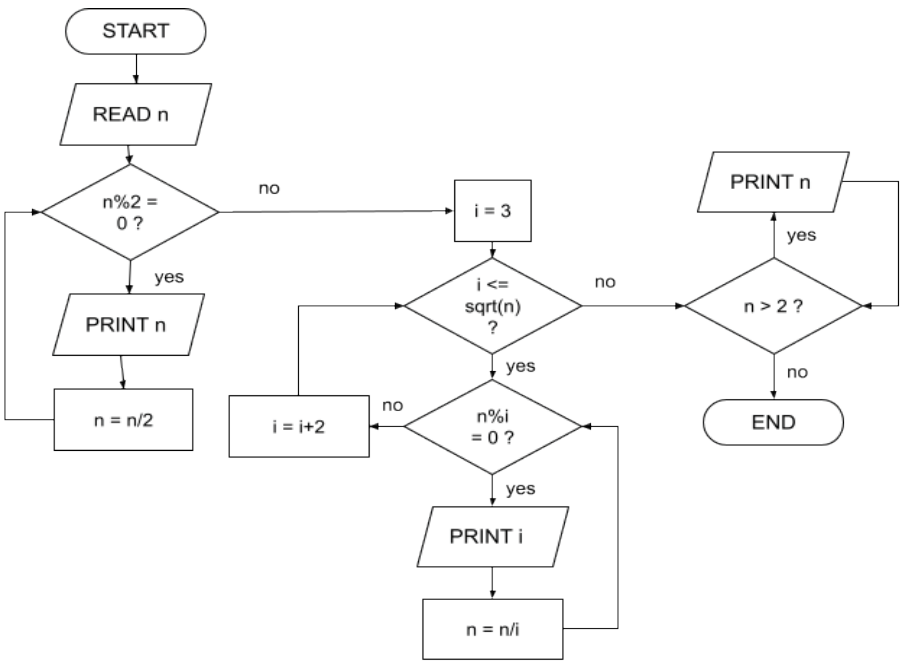
1. (5 Menit) Guru membuka kelas dan dapat melakukan pemanasan.
2. (10 Menit) Guru menjelaskan siswa ke aktivitas Ayo, Kita Berlatih 8: Latihan Pemrograman.
3. (45 Menit) Guru memberikan waktu pada siswa untuk menyelesaikan soal pertama.
4. (35 Menit) Guru memberikan kesempatan bagi siswa untuk menyajikan hasil pekerjaannya.
5. (45 Menit) Guru memberikan waktu pada siswa untuk menyelesaikan soal kedua.
6. (35 Menit) Guru memberikan kesempatan bagi siswa untuk menyajikan hasil pekerjaannya.

7. (45 Menit) Guru memberikan waktu pada siswa untuk menyelesaikan soal kedua.
8. (35 Menit) Guru memberikan kesempatan bagi siswa untuk menyajikan hasil pekerjaannya.
9. (15 Menit) Guru menjelaskan beda koding dengan pemrograman, dan menjelaskan bahwa kemampuan dari unit ini akan digunakan kembali pada unit yang lain, seperti analisis data, dan akan diteruskan di kelas XI. Guru menutup kelas dan mengarahkan siswa pada aktivitas refleksi. Kode hasil pekerjaan dapat dikumpulkan untuk diperiksa.

Pembahasan Ayo, Berlatih 8

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

Diagram Alir:



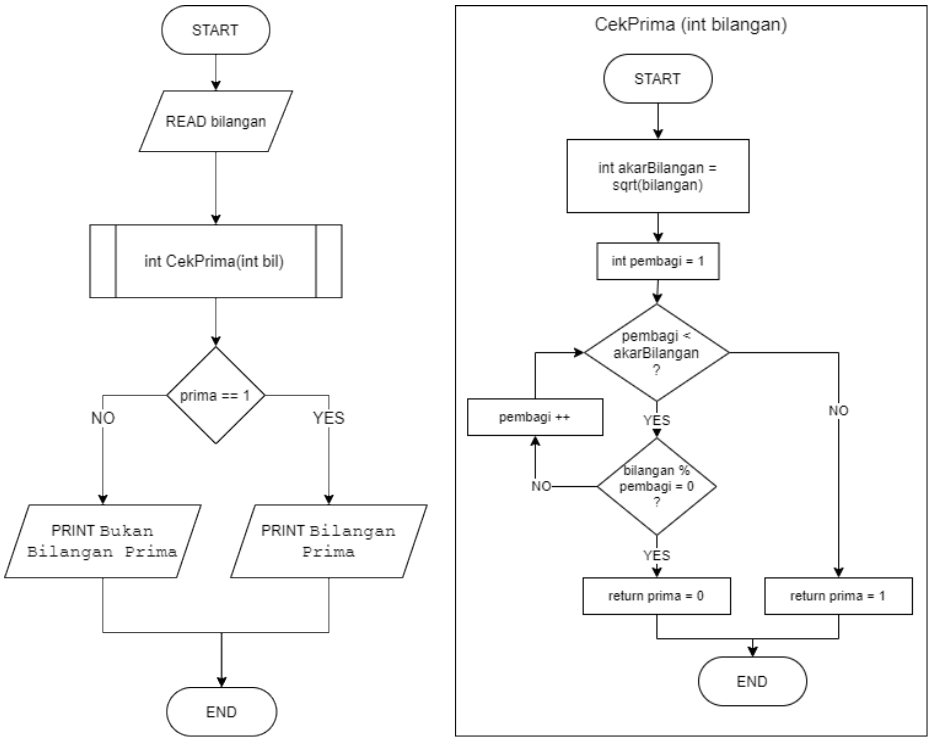
Gambar 7.5 Flowchart Soal 1 Ayo, Berlatih 8
Sumber: Dokumen Kemendikbud, 2021

Contoh Solusi Program:

```
/*
 * Mencetak Faktor Prima
 */

#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);

    // Cetak seluruh faktor dua
    while (n%2 == 0) {
        printf("%d ", 2);
        n = n/2;
    }

    // Cek apakah n habis dibagi bilangan ganjil mulai
    // dari 3
    for (int i = 3; i <= sqrt(n); i = i+2) {

        // Selama n habis dibagi i, cetak i, dan bagi n
         //dengan i
        while (n%i == 0) {
            printf("%d ", i);
            n = n/i;
        }
    }
    // Apabila n > 2 di bagian ini, n pasti prima
    if (n > 2)
        printf ("%d ", n);
    return 0;
}
```

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

Diagram Alir:



Gambar 7.6 Flowchart Soal 2 Ayo, Berlatih 8
Sumber: Dokumen Kemendikbud, 2021

Contoh Solusi Program:

```
/*
 * Mengecek Apakah Bilangan Prima
 */

#include <stdio.h>
#include <math.h>

// Bisa saja tidak dibuat dalam bentuk fungsi
int cekPrima(int bilangan){
      int pembagi;
      int akarBilangan = sqrt(bilangan);
```

```
      // Bagi bilangan dengan pembagi dari 2 hingga akar n
      for(pembagi = 1; pembagi < akarBilangan; pembagi++)
            if(bilangan % pembagi == 0)
                  return 0;
      return 1;
}

int main() {
      int bilangan;
      scanf("%d", &bilangan);

      if(cekPrima(bilangan) == 1)
            printf("Bilangan Prima\n");
      else
            printf("Bukan Bilangan Prima\n");
     return 0;
}
```

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

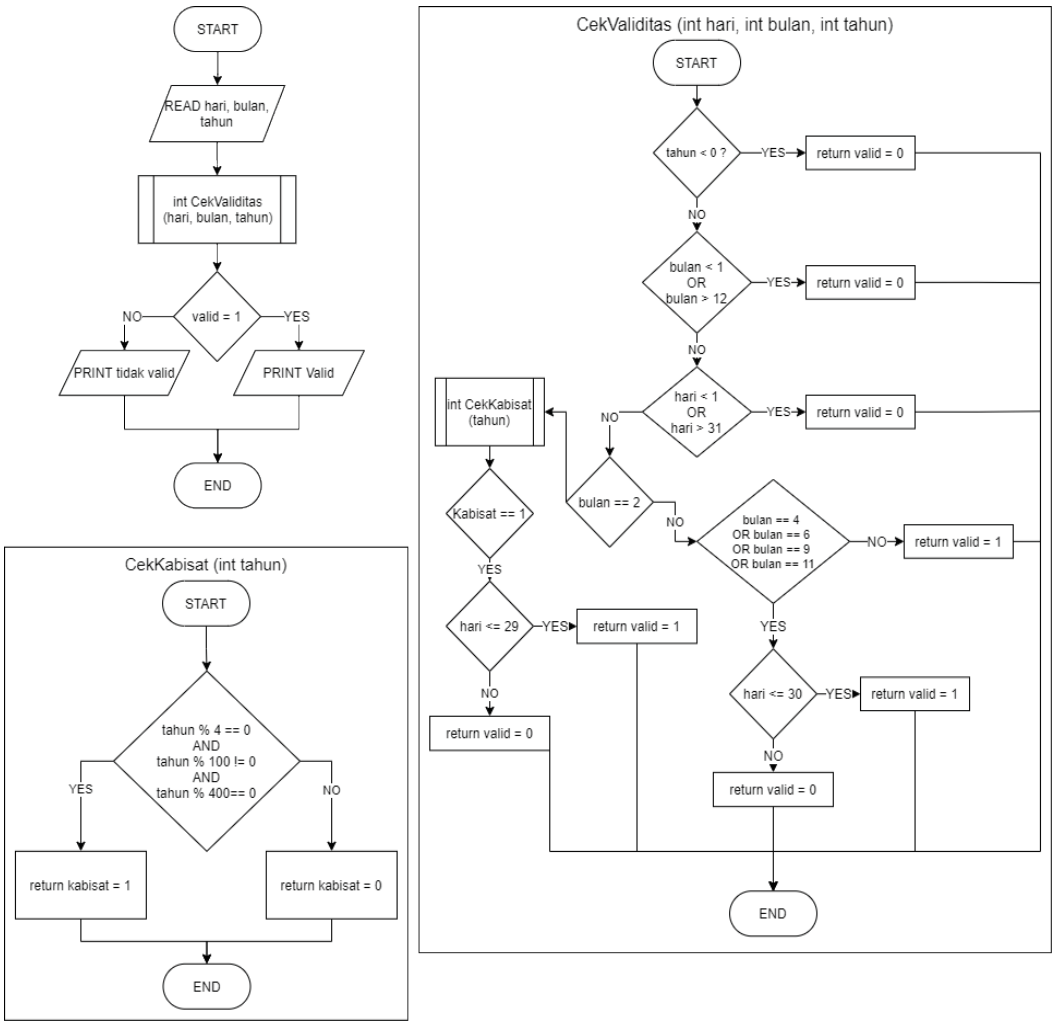
Petunjuk Singkat:

Siswa dapat mulai dengan membahas asumsi dan aturan-aturan berikut tentang validitas suatu tanggal kalender:

1. Tanggal tidak kurang dari 1 dan tidak lebih dari 31 (1 - 31).
2. Bulan tidak kurang dari 1 dan tidak lebih dari 12 (1 - 12).
3. Tahun harus lebih besar dari 0.
4. Saat bulan April, Juni, September, November tanggal tidak lebih dari 30.
5. Saat bulan Februari periksa jika bulan kabisat tanggal tidak lebih dari 29, jika tidak, tanggal tidak lebih dari 28.

Diagram Alir:

Diagram alir ini hanya merupakan salah satu solusi sebab urutan pengecekan dapat dimulai dari hari, bulan kemudian tahun atau urutan lainnya yang menghasilkan diagram alir yang berbeda.



Gambar 7.7 Flowchart Soal 3 Ayo, Berlatih 8
Sumber: Dokumen Kemendikbud, 2021

Contoh Solusi Program:

```
/*
 * Program untuk Mengecek apakah Tanggal valid
 */
#include <stdio.h>
//Fungsi untuk cek apakah sebuah tahun merupakan tahun
kabisat
int cekKabisat(int tahun) {
   if((tahun % 4 == 0) && (tahun % 100 != 0) && (tahun %
   400 == 0))
      return 1;
   else
      return 0;
}
// Fungsi untuk mengecek apakah sebuah tanggal valid
int cekValiditasTanggal(int hari, int bulan, int
tahun) {

   // Cek jangkauan tanggal, bulan, dan tahun
   if(tahun < 0)
      return 0;
   if(bulan < 1 || bulan > 12)
      return 0;
   if(hari < 1 || hari > 31)
      return 0;

   // Cek validitas tanggal di setiap bulan
   if( bulan == 2 ) {
       // Cek bulan februari
      if(cekKabisat(tahun)) {
         if(hari <= 29)
            return 1;
         else
            return 0;
      }
   }
```

```
    // April, June, September, November terdiri atas 30 hari
    if ( bulan == 4 || bulan == 6 || bulan == 9 || bulan
     == 11 ){
       if(hari <= 30)
          return 1;
       else
          return 0;
    }

    // Sisa tanggal lainnya dipastikan valid
    return 1;
}

int main() {
    int hari, bulan, tahun;
    scanf("%d %d %d", &hari, &bulan, &tahun);
    if(cekValiditasTanggal(hari, bulan, tahun))
       printf("Tanggal Valid\n");
    else
       printf("Tanggal Tidak Valid\n");
    return 0;
}
```

H. Pengayaan Aktivitas Utama

Aktivitas pembelajaran bisa dikembangkan dengan mempelajari materi dari situs-situs yang memiliki reputasi bagus, seperti berikut.

Pelajari lebih lanjut mengenai pemrograman berpasangan (*pair programming*):

1. en.wikipedia.org/wiki/Pair_programming
2. agilealliance.org/glossary/pairing/

Pelajari lebih lanjut mengenai penulisan algoritma

cs.wmich.edu/gupta/teaching/cs3310/sp18cs3310web/lecture%20notes%20cs3310/PseudocodeBasics.pdf

Pelajari lebih lanjut mengenai pemrograman dengan menggunakan bahasa C:

1. Deitel P, Deitel H. 2016. C: How to Program Edisi ke-8.
2. Kernighan & Ritchi, "C programming Language"
3. Memulai Pemrograman dengan C: dicoding.com/academies/120
4. Dokumentasi bahasa C: devdocs.io/c

Situs-situs latihan pemrograman dengan menggunakan *auto-grader*:

1. tlx.toki.id
2. spoj.com

I. Asesmen dan Rubrik Penilaian

Asesmen terhadap aktivitas dibagi menjadi tiga kelompok sesuai dengan tujuan dan materi yang diberikan.

Ayo, Berlatih 1-2

Tabel 7.3 Rubrik Penilaian Ayo, Berlatih 1-2

Indikator	Baik	Sedang	Kurang
Keterbacaan algoritma	Algoritma yang dibuat dapat dibaca dengan sangat baik oleh orang lain. Semua langkah dapat dipahami dan tidak menimbulkan ambiguitas.	Algoritma yang dibuat dapat dibaca dengan kurang baik oleh orang lain. Ada beberapa hal yang ambigu.	Algoritma yang dibuat tidak dapat dibaca dengan baik oleh orang lain.
Ketepatan penggunaan simbol	Semuasimbol pada diagram alir digunakan dengan tepat dan benar.	Ada simbol pada diagram alir yang tidak digunakan dengan tepat dan benar.	Banyak simbol pada diagram alir yang tidak digunakan dengan tepat dan benar.
Ketepatan algoritma yang dibuat.	Algoritma yang dibuat dapat menyelesaikan masalah yang diberikan dengan benar.	Algoritma yang dibuat dapat menyelesaikan sebagian masalah yang diberikan dengan benar. Misalnya, ada beberapa skenario masukan yang membuat algoritma mengeluarkan hasil yang salah.	Algoritma yang dibuat tidak dapat menyelesaikan masalah yang diberikan dengan benar.

Ayo, Lakukan 1 – 2

Tabel 7.4 Rubrik Penilaian Ayo, Lakukan 1-2

Indikator	Baik	Sedang	Kurang
Pelaksanaan aktivitas	Semualangkah pada aktivitas dapat direplikasi dengan sempurna oleh siswa.	Ada langkah pada aktivitas yang tidak dapat direplikasi dengan sempurna oleh siswa.	Banyak langkah pada aktivitas yang tidak dapat direplikasi dengan sempurna oleh siswa.
Hasil aktivitas	Siswa mencapai tujuan akhir dari aktivitas.	Siswa mencapai tujuan akhir dari aktivitas, setelah dipandu oleh guru atau siswa lainnya.	Siswa tidak mencapai tujuan akhir dari aktivitas.

Ayo, Berlatih 3 – 7

Tabel 7.5 Rubrik Penilaian Ayo, Berlatih 3-7

Indikator	Baik	Sedang	Kurang
Proses pembuatan program yang baik.	Siswa membuat program melalui proses yang baik: membuat algoritma, menguji program, dll.	Siswa membuat program melalui proses yang kurang baik.	Siswa membuat program melalui proses yang tidak baik.
Praktik baik pemrogram-an.	Siswa mengikuti semuapraktik baik pada pemrograman.	Siswa mengikuti sebagian praktik baik pada pemrograman.	Siswa tidak mengikuti praktik baik pada pemrograman.
Keterbacaan kode program.	Siswa membuat kode yang dapat dibaca dengan baik oleh orang lain.	Siswa membuat kode yang dapat dibaca dengan cukup baik oleh orang lain.	Siswa membuat kode yang sulit dibaca dengan baik oleh orang lain.
Ketepatan program.	Siswa dapat membuat program yang menyelesaikan masalah yang diberikan dengan benar di semua kasus uji yang diberikan.	Siswa dapat membuat program yang menyelesaikan masalah yang diberikan dengan benar di sebagian besar kasus uji yang diberikan.	Siswa dapat membuat program yang menyelesaikan masalah yang diberikan dengan benar di sebagian kecil kasus uji yang diberikan/ program siswa tidak dapat dijalankan.

Integritas	Siswa membuat program dengan jujur dan tidak melakukan plagiasi.	Siswa membuat program dengan adanya plagiasi pada sebagian kecil kode program.	Siswa membuat program dengan adanya plagiasi pada sebagian besar atau semuakode program.

Ayo, Berlatih 8

Tabel 7.6 Rubrik Penilaian Ayo, Berlatih 8

Indikator	Baik	Sedang	Kurang
Konstruksi Algoritma	Siswa memodelkan strategi berpikirnya dalam bentuk algoritma dengan benar dan dapat dipahami.	Siswa memodelkan strategi berpikirnya dalam bentuk algoritma dengan benar dan cukup dipahami.	Siswa memodelkan strategi berpikirnya dalam bentuk algoritma kurang benar dan sulit dipahami.
Proses pembuatan program yang baik.	Siswa membuat program melalui proses yang baik: membuat algoritma, menguji program, dll.	Siswa membuat program melalui proses yang kurang baik.	Siswa membuat program melalui proses yang tidak baik.
Praktik baik pemrogram-an.	Siswa mengikuti semuspraktik baik pada pemrograman.	Siswa mengikuti sebagian praktik baik pada pemrograman.	Siswa tidak mengikuti praktik baik pada pemrograman.
Keterbacaan kode program.	Siswa membuat kode yang dapat dibaca dengan baik oleh orang lain.	Siswa membuat kode yang dapat dibaca dengan cukup baik oleh orang lain.	Siswa membuat kode yang sulit dibaca dengan baik oleh orang lain.
Ketepatan program.	Siswa dapat membuat program yang menyelesaikan masalah yang diberikan dengan benar di semua kasus uji yang diberikan.	Siswa dapat membuat program yang menyelesaikan masalah yang diberikan dengan benar di sebagian besar kasus uji yang diberikan.	Siswa dapat membuat program yang menyelesaikan masalah yang diberikan dengan benar di sebagian kecil kasus uji yang diberikan / program siswa tidak dapat dijalankan.
Integritas	Siswa membuat program dengan jujur dan tidak melakukan plagiasi.	Siswa membuat program dengan adanya plagiasi pada sebagian kecil kode program.	Siswa membuat program dengan adanya plagiasi pada sebagian besar atua seluruh kode program.

J. Interaksi Guru dan Orang Tua/Wali

Orang tua/wali hendaknya selalu aktif dalam mengawasi anaknya ketika melakukan aktivitas online. Guru dapat berinteraksi dengan memberikan informasi dan tips bagi orang tua dalam penggunaan tools pemrograman yang ada sehingga orang tua juga dapat membantu anaknya ketika mengalami kesulitan.