**Template Lembar Kerja Individu dan Kelompok**

|  |  |  |
| --- | --- | --- |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **Sindi Putri Utami**  **G1F024053** | **Operator Java** | **05 September 2024** |
| **[1] Identifikasi Masalah:** | | |
| 1.1.  Tambahkan baris System.out.println("a + b = "  + (a + b));  Ubahlah operator ( + ) dengan tanda ( -, \*, /, %) 1.2.  Analisa perhitungan matematika yang terjadi! | | |
| **[1] Analisis dan Argumentasi** | | |
| 1 .2. analisis perhitungan   1. Penjumlahan (a + b)   perhitungan**:** 20 + 3 = 23  Output: a + b = 23  Menampilkan hasil penjumlahan dari a dan b.   1. Pengurangan (a - b)   Perhitungan**:** 20 - 3 = 17  Output: a - b = 17  Menampilkan hasil pengurangan dari a dan b.   1. Perkalian (a \* b)   Perhitungan: 20 \* 3 = 60  Output: a \* b = 60  Menampilkan hasil perkalian dari a dan b.   1. Pembagian (a / b)   Perhitungan: 20 / 3 = 6 (pembagian integer membulatkan hasil ke bawah)  Output: a / b = 6  Menampilkan hasil pembagian dari a oleh b. (Pembagian dengan tipe data integer).   1. Modulus/sisa bagi (a % b)   Perhitungan: 20 % 3 = 2  Output: a % b = 2  Menampilkan sisa dari pembagian a dengan b. | | |
| **[1] Penyusunan Algoritma dan Kode Program** | | |
| * 1. Evaluasi penyebab kesalahan yang terjadi dan perbaiki agar program dapat berjalan   Penyusunan Algoritma:   1. Mulai 2. Nama file dan kelas 3. Method main 4. Deklarasi nilai a dan b 5. Tambahkan System.out.println 6. Jalankan 7. Selesai | | |
| Output sudah sesuai dengan inputan | | |
| **[1] Kesimpulan** | | |
| * Penjumlahan: Menambahkan dua angka untuk mendapatkan jumlah total. * Pengurangan: Mengurangi satu angka dari angka lainnya untuk mendapatkan selisih. * Perkalian: Mengalikan dua angka untuk mendapatkan hasil perkalian. * Pembagian: Membagi satu angka dengan angka lainnya untuk mendapatkan hasil bagi, dengan hasil dibulatkan ke bawah dalam tipe data integer. * Modulus/sisa bagi: Menentukan sisa dari pembagian dua angka. | | |
| |  | | --- | | **[2] Identifikasi Masalah:** | | 2.1.  Bandingkan hasil Contoh 1 dengan Contoh 2! | | **[2] Analisis dan Argumentasi** | | Analisis contoh 1   * Penambahan (a + b): 20 + 3 = 23. Ini adalah operasi penjumlahan sederhana. * Pengurangan (a - b): 20 - 3 = 17. Ini adalah operasi pengurangan sederhana. * Perkalian (a \* b): 20 \* 3 = 60. Ini adalah operasi perkalian sederhana. * Pembagian (a / b): 20 / 3 = 6. Dalam Java, pembagian antara dua bilangan bulat (integer) menghasilkan bilangan bulat, jadi sisa bagi dibuang. * Modulus/Sisa Bagi (a % b): 20 % 3 = 2. Ini menunjukkan sisa dari pembagian 20 dengan 3.   Argumentasi: Kode ini menunjukkan dasar dari operator aritmatika yang digunakan untuk melakukan berbagai jenis operasi matematika.  Analisis contoh 2   * Penambahan (b += a): 3 + 20 = 23. Nilai b diubah dari 3 menjadi 23. * Pengurangan (b -= a): 23 - 20 = 3. Nilai b diubah dari 23 menjadi 3. * Perkalian (b \*= a): 3 \* 20 = 60. Nilai b diubah dari 3 menjadi 60. * Pembagian (b /= a): 60 / 20 = 3. Nilai b diubah dari 60 menjadi 3. * Sisa Bagi (b %= a): 3 % 20 = 3. Nilai b dari 3 tetap menjadi 3 karena 3 lebih kecil dari 20.   Argumentasi: Operator penugasan (+=, -=, \*=, /=, %=) mengubah nilai variabel yang ada. Setiap operator ini mengubah nilai variabel berdasarkan operasi matematika dan menyimpannya kembali ke variabel tersebut.   * Contoh 1 menunjukkan hasil dari operasi aritmatika dasar tanpa mengubah nilai variabel yang digunakan dalam operasi. * Contoh 2 menunjukkan hasil dari penggunaan operator penugasan, nilai variabel b diubah setelah setiap operasi, mempengaruhi hasil operasi berikutnya. | | **[2] Penyusunan Algoritma dan Kode Program** | | Penyusunan Algoritma:   1. Mulai 2. Nama file dan kelas 3. Method main 4. Deklarasi nilai a dan b 5. Menampilkan hasil dari variable Jalankan 6. Selesai | | Luaran sesuai dengan masukan | | **[2] Kesimpulan** | | Penjumlahan, Pengurangan, Perkalian, Pembagian, dan Modulus(sisa bagi) adalah operasi matematika dasar yang digunakan untuk melakukan perhitungan dengan dua angka.   * Penjumlahan (a + b): Menjumlahkan dua angka untuk mendapatkan hasil total. * Pengurangan (a - b): Mengurangi satu angka dari angka lainnya untuk mendapatkan selisih. * Perkalian (a \* b): Mengalikan dua angka untuk mendapatkan hasil. * Pembagian (a / b): Membagi satu angka dengan angka lainnya, menghasilkan hasil bagi dengan pembulatan ke bawah dalam tipe data integer. * Modulus/sisa bagi (a % b): Menentukan sisa dari pembagian dua angka.   Semua operasi mengikuti aturan matematika dasar dan memberikan hasil yang sesuai dengan harapan berdasarkan tipe data yang digunakan. | | **[3] Identifikasi Masalah:** | | 3.1.  Ubahlah nilai A = 4 dan B = 4. Analisa perubahan yang terjadi!  3.2   Bandingkan bagaimana perbedaan nilai A dan B mempengaruhi nilai luaran! | | **[3] Analisis dan Argumentasi** | | 3.1. Analisa perubahan.   1. Operator > (lebih besar dari):  * Sebelumnya (A = 12, B = 4): A > B menghasilkan true karena 12 lebih besar dari 4. * Sekarang (A = 4, B = 4): A > B menghasilkan false karena 4 tidak lebih besar dari 4.  1. Operator < (lebih kecil dari):  * Sebelumnya (A = 12, B = 4): A < B menghasilkan false karena 12 tidak lebih kecil dari 4. * Sekarang (A = 4, B = 4): A < B menghasilkan false karena 4 tidak lebih kecil dari 4.  1. Operator >= (lebih besar atau sama dengan):  * Sebelumnya (A = 12, B = 4): A >= B menghasilkan true karena 12 lebih besar atau sama dengan 4. * Sekarang (A = 4, B = 4): A >= B menghasilkan true karena 4 sama dengan 4, jadi juga memenuhi syarat lebih besar atau sama dengan.  1. Operator <= (lebih kecil atau sama dengan):  * Sebelumnya (A = 12, B = 4): A <= B menghasilkan false karena 12 tidak lebih kecil atau sama dengan 4. * Sekarang (A = 4, B = 4): A <= B menghasilkan true karena 4 sama dengan 4, jadi juga memenuhi syarat lebih kecil atau sama dengan.  1. Operator == (sama dengan):  * Sebelumnya (A = 12, B = 4): A == B menghasilkan false karena 12 tidak sama dengan 4. * Sekarang (A = 4, B = 4): A == B menghasilkan true karena 4 sama dengan 4.  1. Operator != (tidak sama dengan):  * Sebelumnya (A = 12, B = 4): A != B menghasilkan true karena 12 tidak sama dengan 4. * Sekarang (A = 4, B = 4**):** A != B menghasilkan false karena 4 sama dengan 4, jadi tidak memenuhi syarat tidak sama dengan.   3.2. Perubahan nilai A dan B mempengaruhi hasil dari setiap operator relasional sebagai berikut:   * Perbandingan > dan <: Hasil dari operator ini tergantung pada nilai A lebih besar atau lebih kecil dari B. Ketika A dan B sama, kedua perbandingan ini akan false. * Perbandingan >= dan <=: Hasil dari operator ini tergantung pada nilai A lebih besar atau sama dengan B, atau lebih kecil atau sama dengan B. Ketika A dan B sama, kedua perbandingan ini akan true. * Perbandingan == dan !=: Hasil dari == adalah true jika nilai A sama dengan B, dan != adalah true jika nilai A tidak sama dengan B. Ketika A dan B sama, == akan true dan != akan false. | | **[3] Penyusunan Algoritma dan Kode Program** | | * Penyusunan Algoritma:   1. Mulai   2. Nama file dan kelas   3. Method main   4. Deklarasi nilai a dan b   5. Perbandingan relasional   6. Selesai     Luaran sudah sesuai dengan masukan | | **[3] Kesimpulan** | | * Ketika A lebih besar dari B, operator > memberikan true dan < memberikan false. Sebaliknya, ketika A sama dengan B, hasil dari > dan < akan false, sedangkan >= dan <= akan true. * Perbandingan == dan != sangat dipengaruhi oleh kesamaan nilai. Jika A dan B sama, == akan true dan != akan false. |  |  | | --- | | **[4] Identifikasi Masalah:** | | 4.1.  Berdasarkan luaran program Contoh 4, bandingkan hasil Post dan Pre untuk Increment dan Decrement! | | **[4] Analisis dan Argumentasi** | | 4.1. Perbandingan Hasil  1. Post-Increment (a++)   * Sebelum Increment**:** a = 10 * Output **a++:** Menampilkan nilai a sebelum increment yaitu 10. * Setelah Increment**:** Nilai a berubah menjadi 11.   Penjelasan:   * **a++** pertama menampilkan nilai variabel a saat ini (10), kemudian menambah nilai a setelah baris tersebut, sehingga a menjadi 11.   **2**. Pre-Increment **(++b)**   * Sebelum Increment**:** b = 10 * Output **++b:** Menampilkan nilai b setelah increment yaitu 11. * Setelah Increment**:** Nilai b tetap 11.   Penjelasan:   * **++b** terlebih dahulu menambah nilai variabel b sebelum menampilkan hasilnya. Jadi, nilai b menjadi 11 sebelum ditampilkan. | | **[4] Penyusunan Algoritma dan Kode Program** | | * Penyusunan Algoritma:   1. Mulai   2. Nama file dan kelas   3. Method main   4. **Tampilkan Nilai Awal** dari a   5. **ampilkan Nilai Post-Increment** dari a   6. **Tampilkan Nilai Baru** dari a   7. Selesai         Luaran sesuai dengan masukkan | | **[4] Kesimpulan** | |  Post-Increment (a++) Menampilkan nilai variabel saat ini, lalu menambah 1.  Contoh: Jika a = 10, maka a++ menampilkan 10 dulu, kemudian a berubah menjadi 11.   Pre-Increment (++b) Menambah 1 pada variabel terlebih dahulu, lalu menampilkan nilai yang baru.  Contoh: Jika b = 10, maka ++b menambah b jadi 11 sebelum menampilkan nilainya, sehingga menampilkan 11.   Post-Decrement (c--) Menampilkan nilai variabel saat ini, lalu mengurangi 1.  Contoh: Jika c = 10, maka c-- menampilkan 10 dulu, kemudian c berubah menjadi 9.   Pre-Decrement (--d) Mengurangi 1 pada variabel terlebih dahulu, lalu menampilkan nilai yang baru.  Contoh: Jika d = 10, maka --d mengurangi d jadi 9 sebelum menampilkan nilainya, sehingga menampilkan 9.   |  | | --- | |  |  |  | | --- | | **[5] Identifikasi Masalah:** | | 5.1.  Tambahkan baris kode untuk memeriksa a || b. 5.2.  Ubahlah nilai a = false dan b = false. Analisa perubahan dan perbedaan boolean yang terjadi! 5.3.  Apabila diketahui pernyataan a || b && a || !b. Uraikan urutan logika yang akan dikerjakan! Analisa luaran true atau false dari pernyataan tersebut! | | **[5] Analisis dan Argumentasi** | | 5.1. Menambahkan Pemeriksaan a || b   * **Operasi** a && b**:** Operasi AND (&&) hanya menghasilkan true jika kedua operand adalah true. Dalam hal ini, karena b adalah false, hasilnya adalah false. * **Operasi** a || b**:** Operasi OR (||) menghasilkan true jika salah satu dari operandnya adalah true. Karena a adalah true, hasilnya adalah true.   **Argumentasi:**   * **AND (**&&**):** Memeriksa bahwa kedua kondisi harus benar untuk menghasilkan true. * **OR (**||**):** Memeriksa bahwa setidaknya satu kondisi harus benar. Ini berguna untuk situasi di mana hanya salah satu dari beberapa kondisi yang perlu dipenuhi.   1. Mengubah Nilai a dan b Menjadi false * Operasi a && b Dengan kedua operand (a dan b) adalah false, hasilnya adalah false karena AND memerlukan kedua operand untuk menjadi true. * Operasi a || b: Dengan kedua operand adalah false. hasilnya adala false karena OR memerlukan setidaknya salah satu operand untuk menjadi true.   Argumentasi:   * AND (&&): Operasi ini digunakan untuk memverifikasi bahwa semua kondisi harus benar. Jika salah satu kondisi adalah false, hasilnya adalah false. * OR (||): Operasi ini digunakan untuk memeriksa apakah ada satu kondisi yang benar. Jika semua kondisi adalah false , hasilnya adalah false.   1. Analisis Pernyataan a || b && a ||  1. Evaluasi !b:   Dengan b = false, !b (NOT b) adalah true.   1. Evaluasi b && a:   Dengan b = false dan a = true, b && a (AND) menghasilkan false karena salah satu operand adalah false.   1. Evaluasi a || (b && a):   Dengan hasil b && a adalah false, maka a || false menghasilkan true karena a adalah true.   1. Evaluasi (a || (b && a)) || !b:   Dengan a || (b && a) menghasilkan true dan !b adalah true, maka true || true adalah true.  Argumentasi:   * Urutan Evaluasi: Operasi && memiliki prioritas lebih tinggi dibandingkan dengan ||, dan ! memiliki prioritas tertinggi. Oleh karena itu, !b dievaluasi terlebih dahulu, diikuti oleh b && a, kemudian a || (b && a), dan akhirnya a || (b && a) || !b. * Hasil Akhir: Hasil akhir adalah true karena kombinasi dari kondisi yang benar (a dan !b) membuat keseluruhan pernyataan benar. | | **[5] Penyusunan Algoritma dan Kode Program** | | 5.1. Penyusunan Algoritma:   * 1. Mulai   2. Nama file dan kelas   3. Method main   4. Nilai variable a dan b   5. Tambahkan baris kode untuk a || b.   6. **Jalankan**   7. Selesai     Luaran sudah sesua dengan masukkan  5.2. Penyusunan Algoritma:   * 1. Mulai   2. Nama file dan kelas   3. Method main   4. Nilai variable a dan b   5. Ubah nilai a dan b menjadi false dan false   6. **Jalankan**   7. Selesai     Luaran sudah sesuai dengan maskkan   * + Penyusunan Algoritma:   1. Mulai   2. Nama file dan kelas   3. Method main   4. Nilai variable a dan b   5. **Jalankan**   6. Selesai     Luaran sudah sesuai dengan masukkan | | **[5] Kesimpulan** | | Dalam kedua kasus, hasil akhir dari pernyataan a || b && a || !b adalah true. Karena, salah satu dari elemen || akan bernilai true, yang membuat keseluruhan ekspresi menjadi true. | | **[6] Identifikasi Masalah:** | | Berdasarkan Contoh 6, ubahlah nilai = 60. Analisis hasil dan proses yang terjadi! | | **[6] Analisis dan Argumentasi** | | * Variabel:   String status = ""; : Kita buat variabel bernama status yang awalnya kosong.  int nilai = 80; : Kita beri nilai 80 pada variabel nilai.   * Cek Kondisi:   status = (nilai > 60) ? "Lulus" : "Gagal"; : Bagian ini adalah cara singkat untuk memeriksa kondisi dan mengatur nillai status.  Jika nilai lebih besar dari 60, status akan menjadi "Lulus". Jika nilai tidak lebih besar dari 60, status akan menjadi "Gagal".   * Hasil:   Karena 80 lebih besar dari 60, hasilnya adalah "Lulus". Program akan mencetak "Lulus". | | **[6] Penyusunan Algoritma dan Kode Program** | | Penyusunan Algoritma:   * 1. Mulai   2. Nama file dan kelas   3. Method main   4. Deklarasi variable   5. **Cek kondisi**   6. Tentukan Status   7. **Jalankan**   8. Selesai     Luaran sesuai dengan masukkan | | **[6] Kesimpulan** | | * Saat nilai lebih dari 60 (misalnya 80), hasilnya adalah "Lulus". * Saat nilai sama dengan 60 atau kurang dari 60, hasilnya adalah "Gagal".   Ini karena kondisi nilai > 60 hanya benar jika nilai benar-benar lebih besar dari 60. | | **[7] Identifikasi Masalah:** | | Pilihlah 3 perhitungan Contoh 7, kemudian uraikan perhitungan biner! Simpulkan hasilnya! | | **[7] Analisis dan Argumentasi** | | * Inisialisasi Variabel:   int a = 10;  int b = 7;  int hasil;   * Bitwise & (AND)   Desimal: 2  Penjelasan: Operator & melakukan AND bitwise pada setiap bit dari kedua angka. Hasilnya adalah 1 hanya jika kedua bit adalah 1, jika tidak hasilnya 0. Dalam hal ini, hanya bit ke-2 dari kanan yang hasilnya 1, sehingga hasil keseluruhannya adalah 2.   * + - Bitwise |(OR):   Operator | melakukan OR bitwise pada setiap bit dari kedua angka. Hasilnya adalah 1 jika salah satu dari kedua bit adalah 1. Dalam hal ini, semua bit menjadi 1, sehingga hasil keseluruhannya adalah 15.   * + - Bitwise ^ (XOR)   Operator ^ melakukan XOR bitwise pada setiap bit dari kedua angka. Hasilnya adalah 1 jika bit-bit yang dibandingkan berbeda, dan 0 jika bit-bitnya sama. Dalam hal ini, bit-bit yang berbeda menghasilkan 1 di posisi ke-3, ke-4, dan ke-5 dari kanan, sehingga hasil keseluruhannya adalah 13. | | **[7] Penyusunan Algoritma dan Kode Program** | | Penyusunan Algoritma:   * 1. Mulai   2. Nama file dan kelas   3. Method main   4. Nilai variable a dan b   5. **Jalankan**   6. Selesai     Luaran sesuai dengan masukkan | | **[7] Kesimpulan** | | * + - 1. AND (&): Menghasilkan 1 jika kedua bit pada posisi yang sama adalah 1, sebaliknya 0.   2. OR (|): Menghasilkan 1 jika salah satu dari kedua bit pada posisi yang sama adalah 1, sebaliknya 0.  3. XOR (^): Menghasilkan 1 jika bit pada posisi yang sama berbeda, sebaliknya 0.  Operasi bitwise adalah teknik penting dalam pemrograman yang memungkinkan manipulasi data pada level bit. Sering digunakan dalam situasi yang memerlukan efisiensi. Pemahaman yang mendalam tentang operasi bitwise membantu dalam optimasi dan pemecahan masalah yang melibatkan manipulasi data bit-level. |   Refleksi:  Ketika baru belajar, merasa bingung dengan berbagai jenis operator. seiring waktu dan latihan,akan semakin terbiasa. | | | |