Adventure of Algorithms

Muhamad Alfauzi Syahputra
2509106006

Anugerah Fakhriza Reswara
2509106025

Bab 1: Jalan Buntu Pertama dan Logika Logam

Elara menghela napas panjang, uap hangat dari bibirnya langsung lenyap ditelan kabut Lembah Kelabu. Selubung putih pekat itu membungkusnya seperti kain basah, meredam suara langkah kakinya sendiri dan mengubah pepohonan menjadi hantu-hantu kelabu di kejauhan. Ini bukan kali pertama ia tersesat. Sejak kecil, rasa ingin tahunya yang besar selalu membawanya ke tempat-tempat tak terduga, tetapi arahnya yang payah sering kali mengubah petualangan menjadi perjuangan untuk bertahan hidup.

Namun, kali ini berbeda. Di pinggangnya tergantung sebuah benda pusaka keluarga Logika Logam. Benda itu bukanlah pedang atau tongkat sihir, melainkan sebuah artefak kuno berbentuk lempengan perunggu seukuran telapak tangan. Konon, permukaannya hanya akan menampilkan tulisan bercahaya bagi mereka yang berpikir jernih dan terstruktur. Bagi Elara, benda itu adalah kompas terbaiknya di tengah badai kebingungan.

Masalah utamanya saat ini adalah ketidakpastian. Ia tidak tahu di mana ia berada, berapa lama lagi harus berjalan, atau apakah persediaannya cukup. Untuk melawan kekacauan itu, ia berhenti dan duduk di atas batu berlumut. Dengan gerakan mantap, ia mengeluarkan Logika Logam. "Baiklah," bisiknya, suaranya terdengar lirih di tengah kesunyian. "Mari kita pecah masalah ini."

Permukaan perunggu itu mulai berpendar redup saat Elara memusatkan pikirannya pada data pertama yang paling krusial persediaan. Tangannya merogoh kantong, menghitung setiap bungkus ransum yang tersisa dengan saksama. Tujuh. Angka itu muncul di benaknya, dan seketika, cahaya di permukaan logam membentuk simbol yang tegas dan utuh *7*. Sebuah bilangan bulat, pasti dan tidak bisa diganggu gugat. Tidak mungkin ada setengah ransum.

Selanjutnya, ia melirik obor di tangannya. Tingkat bahan bakarnya sulit diukur secara presisi. Ia hanya bisa memperkirakan isinya tersisa kurang dari setengah. Seiring pemikirannya, Logika Logam merespons dengan menampilkan deretan angka yang lebih cair *0.45*. Sebuah bilangan pecahan, representasi dari sebuah perkiraan yang jujur.

Pikirannya kemudian melayang mundur, mencari nama lokasi terakhir yang ia ingat sebelum kabut turun. Sebuah nama yang ia berikan sendiri dalam benaknya. "Jurang Lupa." Begitu nama itu terbentuk, serangkaian aksara bercahaya yang indah terukir di permukaan artefak, seolah menangkap gema dari ingatannya "*Jurang Lupa*".

Terakhir, sebuah pertanyaan sederhana yang butuh jawaban ya atau tidak. Cincin Hangat di jarinya, apakah masih aktif? Ia merasakannya dengan saksama. Ada hawa hangat samar yang menjalar, menandakan sihirnya bekerja. Logika Logam menangkap kepastian itu dan menampilkannya dalam satu kata yang bersinar paling terang *True*.

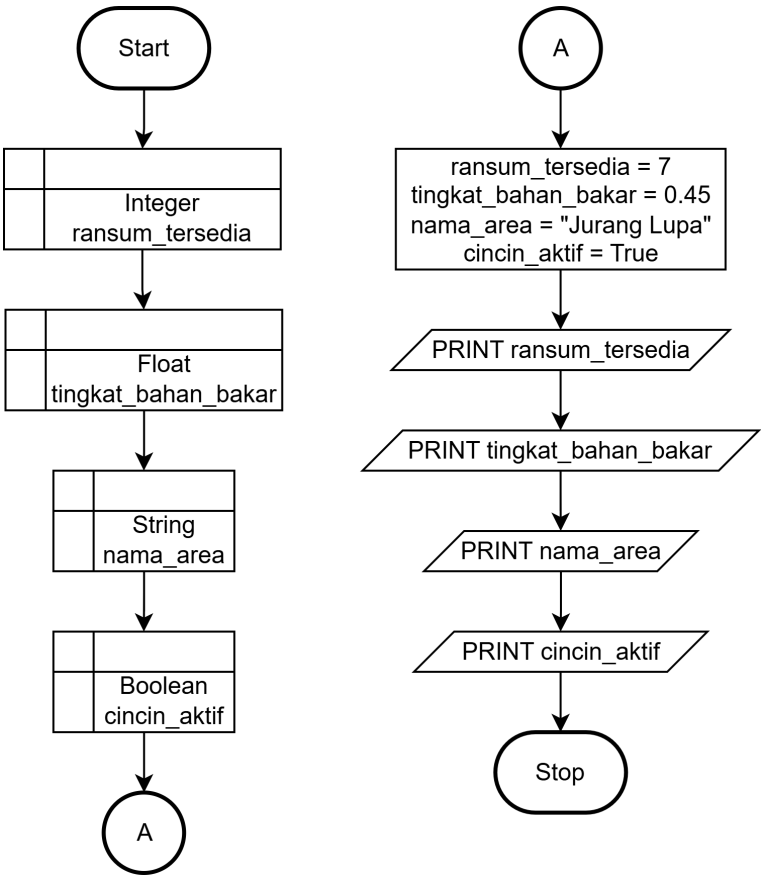
Dengan empat variabel yang kini terdefinisi dengan jelas di hadapannya sebuah integer, float,

string, dan boolean kekacauan di kepala Elara mulai sirna. Kabut di dalam pikirannya perlahan terangkat, digantikan oleh keteraturan data yang memberinya pijakan untuk berpikir langkah selanjutnya. Data-data itu kini tersusun rapi di benak Elara, memberinya pijakan logika pertama. Kabut masih tebal, tetapi jalan di dalam pikirannya mulai terlihat terang.

Pseudocode:

Judul: Logika logam
Kamus / Deklarasi: Integer ransum_tersedia Float tingkat_bahan_bakar = 0.45 String nama_area = “Jurang Lupa” Boolean cincin_aktif = True
Algoritma: PRINT “Status Persediaan Elara:” PRINT ransum_tersedia PRINT tingkat_bahan_bakar PRINT nama_area PRINT cincin_aktif

Flowchart:



Source Code:

```
ransum_tersedia = 7
tingkat_bahan_bakar = 0.45
nama_area = "Jurang Lupa"
cincin_aktif = True
print("Status Persediaan Elara:")
print("Ransum (Integer):", ransum_tersedia)
print("Bahan Bakar Obor (Float):", tingkat_bahan_bakar)
print("Lokasi (String):", nama_area)
print("Cincin Hangat Aktif (Boolean):", cincin_aktif)
```

Output Program:

```
Status Persediaan Elara:
Ransum (Integer): 7
Bahan Bakar Obor (Float): 0.45
Lokasi (String): Jurang Lupa
Cincin Hangat Aktif (Boolean): True
```

Gambar 1.1 Output / Hasil Program

Bab 2: Keputusan di Jembatan Patahan

Elara berjalan kaki selama tiga jam sebelum kabut tipis terangkat. Di depannya terbentang jurang yang lebih kecil, yang dulunya diseberangi oleh Jembatan Gantung Perintis. Kini, jembatan itu patah di tengah, meninggalkan celah selebar tiga meter.

"Melompat atau mencari jalan memutar?" Tangan Elara secara refleks menggenggam Logika Logam, dinginnya perunggu terasa menenangkan di tengah kebingungannya. Ia tahu artefak ini tidak merespons emosi, melainkan struktur. Ia menarik napas dalam-dalam, memproyeksikan data mentah dari benaknya ke permukaan logam simbol angka '7' yang solid muncul untuk ransumnya, di samping simbol '0.45' yang berkilauan cair untuk bahan bakarnya. Keduanya menjadi jangkar data untuk algoritma yang akan ia jalankan.

Pikirannya membentuk perintah pertama yang paling kritis, sebuah uji kelayakan untuk tindakan paling berisiko. "Skenario pertama: situasi mendesak," batinnya, mendefinisikan kondisi itu kepada artefak: *JIKA (ransum < 5) DAN (bahan bakar > 0.3).*

Permukaan artefak itu langsung berdenyut. Sirkuit cahaya tipis terbentuk, menghubungkan kedua jangkar data ke serangkaian logika. Simbol '7' meluncur menuju perbandingan *< 5*. Seketika, koneksi itu berkedip merah kusam False. Karena operator AND membutuhkan semua kondisi bernilai True, seluruh sirkuit untuk skenario "Melompat" itu langsung padam. Jalan pintas yang berisiko itu ditolak oleh logika bahkan sebelum kondisi kedua sempat diuji.

Kegagalan jalur pertama tidak membuatnya ragu itu adalah bagian dari proses. Artefak itu secara otomatis siap untuk evaluasi berikutnya. "Baiklah," pikir Elara, menuntunnya ke logika sekunder. "Jika tidak mendesak, periksa kondisi keamanan apakah bahan bakar masih cukup?" Perintahnya sederhana JIKA *(bahan bakar > 0.3).*

Sirkuit cahaya baru, kali ini berwarna biru es, terbentuk dari simbol '0.45'. Simbol itu meluncur anggun menuju perbandingan *> 0.3*. Saat keduanya bertemu, sambungan itu menyala terang dengan cahaya biru yang mantap True. Sebuah jalur yang valid telah ditemukan. Arus cahaya biru itu mengalir deras, mengabaikan jalur logika lainnya, dan menyalakan serangkaian aksara yang sudah menunggu.

Jalur logika ketiga skenario ELSE untuk kondisi bahan bakar kritis tetap gelap, tidak relevan karena sebuah jawaban telah ditemukan. Semua sirkuit perhitungan yang rumit dan perbandingan yang tidak terpakai memudar menjadi kegelapan.

Yang tersisa hanyalah hasil akhir, bersinar dengan otoritas tenang dalam cahaya biru di permukaan perunggu: "Keputusan: Tidak perlu lompat. Energi obor cukup, cari jalan memutar yang aman."

Cahaya itu bertahan sejenak, memberikan Elara kepastian mutlak. Kemudian, dengan pendaran terakhir, permukaan Logika Logam kembali diam. Getaran logisnya berhenti. Dari tiga kemungkinan masa depan, artefak itu telah menunjukkan satu-satunya jalan yang paling

masuk akal.

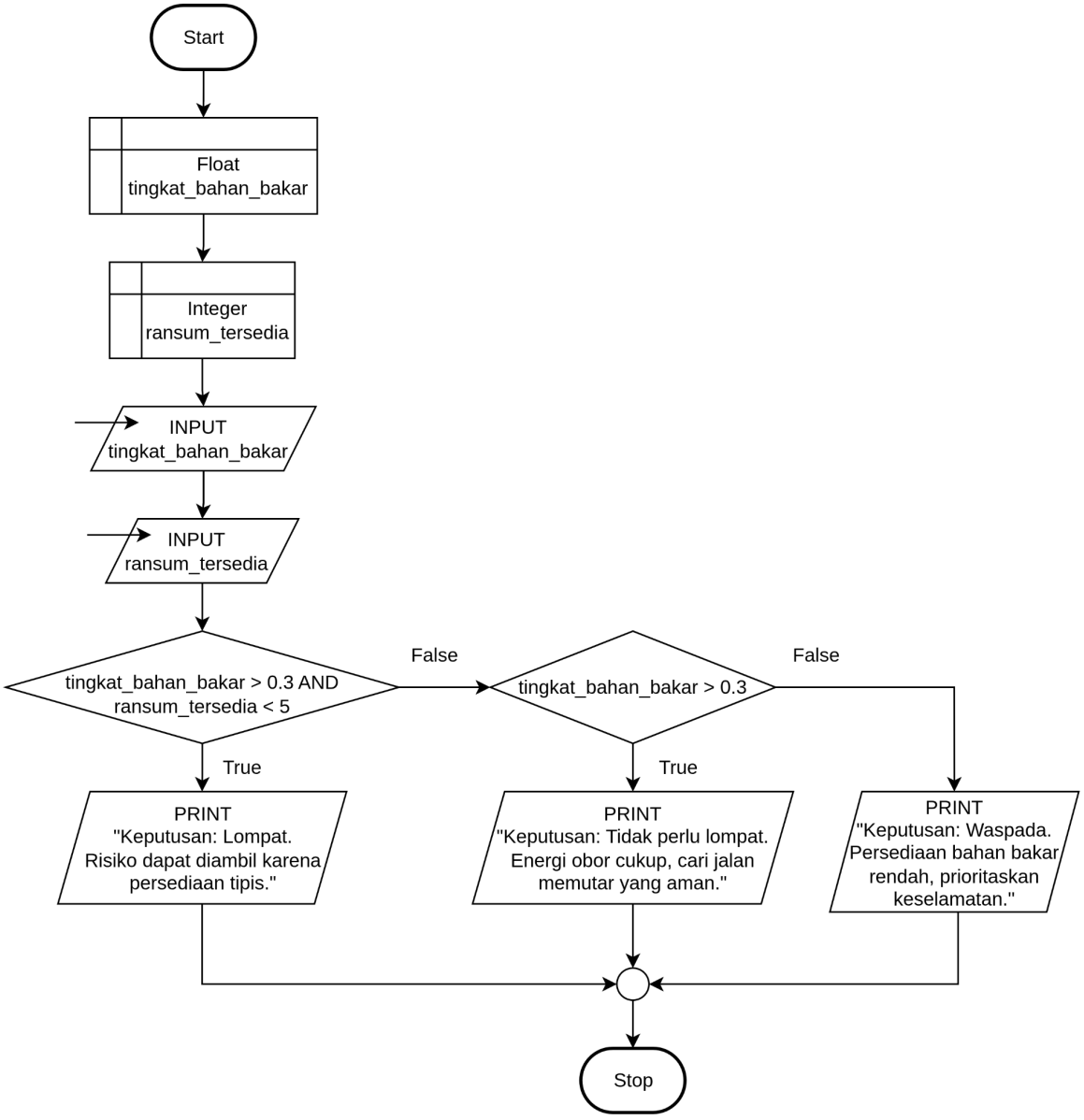
"Jalan memutar," putus Elara, menyimpan Logika Logam. Ransumnya masih cukup banyak, tidak ada alasan untuk mengambil risiko bodoh. Algoritma telah menyelamatkannya dari keputusan impulsif yang bisa berakibat fatal.

Pseudocode:

Judul: Memilih rute menggunakan IF - ELIF - ELSE
Kamus / Deklarasi: Float tingkat_bahan_bakar Integer ransum_tersedia

Algoritma:
INPUT tingkat_bahan_bakar
INPUT ransum_tersedia
IF tingkat_bahan_bakar > 0.3 AND ransum_tersedia < 5 Maka THEN
  PRINT "Lompat. Risiko Dapat Diambil Karena Persediaan Tipis."
ELIF tingkat_bahan_bakar > 0.3 THEN
  PRINT "Tidak Perlu Lompat. Energi Obor Cukup, Cari Jalan Memutar."
ELSE
  PRINT "Waspada. Persediaan Bahan Bakar Rendah, Prioritaskan Keselamatan."

Flowchart:



Source Code:

```
tingkat_bahan_bakar = float(input("tingkat bahan bakar :"))
ransum_tersedia = int(input("masukan ransum tersedia :"))

if tingkat_bahan_bakar > 0.3 and ransum_tersedia < 5:
    print("Keputusan: Lompat. Risiko dapat diambil karena persediaan tipis.")
elif tingkat_bahan_bakar > 0.3:
    print("Keputusan: Tidak perlu lompat. Energi obor cukup, cari jalan memutar
yang aman.")
else:
    print("Keputusan: Waspada. Persediaan bahan bakar rendah, prioritaskan
keselamatan.")
```

Output Program:

```
tingkat bahan bakar :0.2
masukan ransum tersedia :2
Keputusan: Waspada. Persediaan bahan bakar rendah, prioritaskan keselamatan.
```

Gambar 2.1 Kondisi jika Persediaan bahan bakar rendah

```
tingkat bahan bakar :0.45
masukan ransum tersedia :7
Keputusan: Tidak perlu lompat. Energi obor cukup, cari jalan memutar yang aman.
```

Gambar 2.2 Kondisi persedian elera sekarang

Bab 3: Ujian Kebugaran dengan Perulangan

Jalan memutar itu bukanlah sebuah jalan setapak, melainkan dinding hijau yang nyaris vertikal. Lereng curam itu ditutupi lumut tebal seperti karpet basah, menyembunyikan bebatuan licin di bawahnya. Setiap pegangan tampak meragukan, dan setiap pijakan terasa seperti pertaruhan. Naluri pertama Elara adalah bergegas, mencoba mencapai puncak secepat mungkin untuk keluar dari posisi berbahaya ini. Namun, ia segera menahan diri. Ketergesa-gesaan adalah musuh logika, dan di medan seperti ini, musuh itu bisa berakibat fatal.

Tangannya menyentuh Logika Logam di pinggangnya. Artefak itu tidak perlu diaktifkan; prinsipnya sudah terpatri di benaknya. Masalah ini bukanlah tentang kekuatan, melainkan tentang manajemen sumber daya dalam hal ini, energinya sendiri. Kelelahan adalah variabel yang harus dikendalikan.

Maka, ia menciptakan sebuah algoritma untuk tubuhnya. Sebuah perulangan sederhana. Ia akan mendaki dalam satu set yang terdiri dari 25 langkah angka yang cukup untuk membuat kemajuan, tetapi tidak cukup untuk membuatnya kelelahan. Setelah setiap 5 langkah dalam set itu, ia akan berhenti sejenak untuk menarik napas dalam-dalam, sebuah *IF* dalam siklus *WHILE*-nya. Begitu ia menyusun aturan ini di kepalanya, ia merasakan getaran halus dari Logika Logam, sebuah denyut ritmis yang seakan memberinya tempo.

"Baiklah," bisiknya. "Mulai perulangan."

Langkah pertama terasa berat. Otot betisnya langsung menegang. Satu. Ia mencari pijakan lain, jemarinya mencengkeram akar yang menonjol. Dua. Napasnya mulai memburu. Tiga. Ia mengabaikan keinginan untuk melihat seberapa jauh lagi ke puncak. Empat. Fokus hanya pada langkah berikutnya. Lima. Sesuai aturan, ia berhenti, membiarkan paru-parunya terisi penuh. *(Menarik napas sejenak)*. Proses itu berlanjut, monoton dan melelahkan. Hitungan di kepalanya menjadi sebuah mantra yang menenggelamkan rasa lelah dan keraguan. Denyut dari Logika Logam membantunya menjaga ritme, mengubah pendakian yang kacau menjadi serangkaian perintah yang teratur dan efisien.

...Dua puluh empat. Kakinya gemetar. Dua puluh lima. Ia mencapai batasnya. Sesuai algoritma, perulangan berhenti. Ia menekan tubuhnya ke dinding lereng, napasnya terengah-engah. Keringat menetes dari dahi Elara, namun rasa puas memenuhi hatinya. Ia tidak hanya mendaki ia telah mengeksekusi sebuah program dengan sempurna. Logika berhasil mengalahkan kelelahan tanpa batas.

Ia beristirahat, minum sedikit air, dan merasakan kekuatan perlahan kembali. Puncak memang belum terlihat, tetapi ia tahu jalan untuk mencapainya. Setelah beberapa menit, ia siap. Pikirannya kembali fokus, siap memulai iterasi perulangan berikutnya.

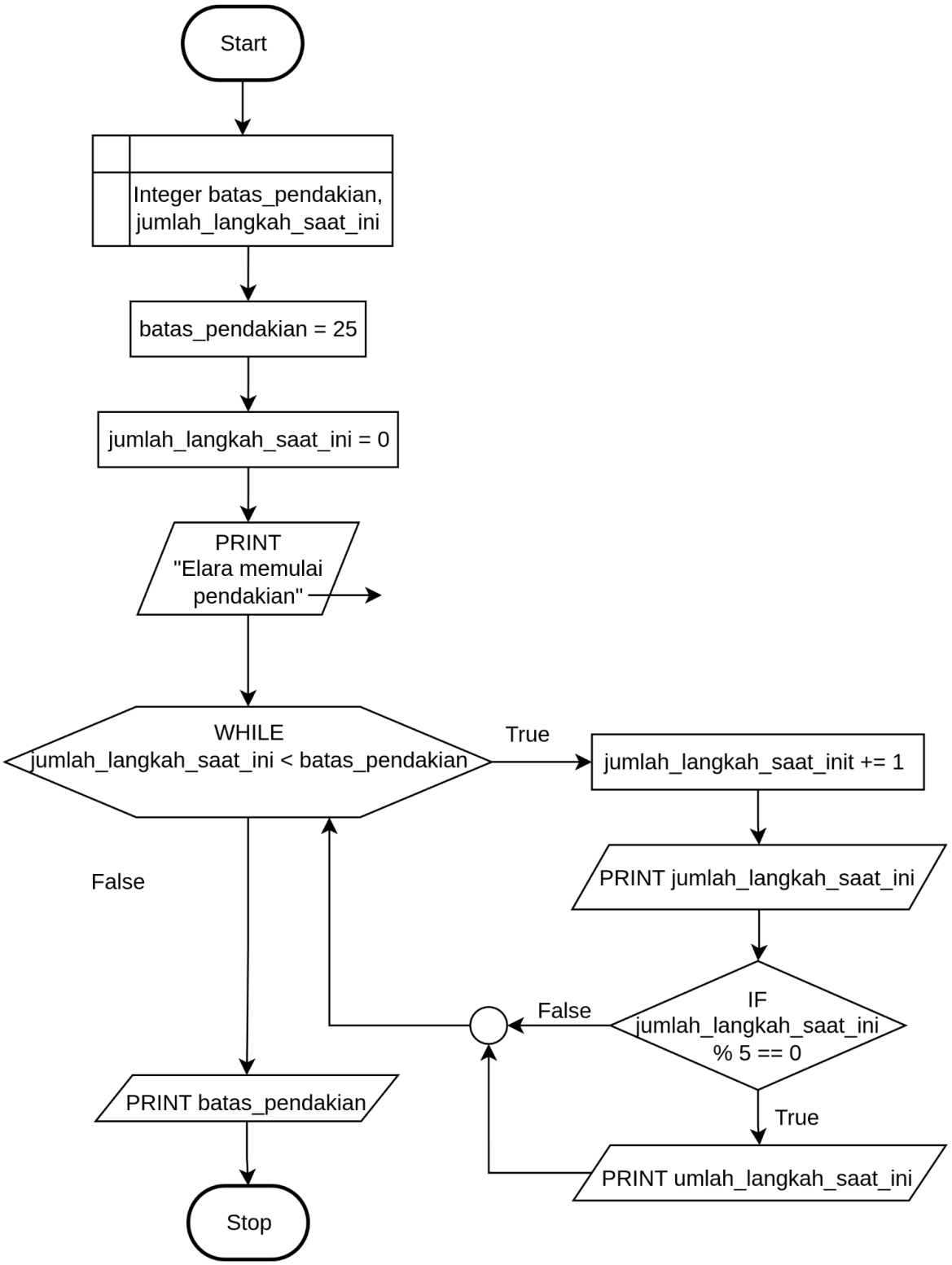
Pseudocode:

Judul :
Mendaki menggunakan perulangan while

Kamus / Deklarasi:
Integer batas_pendakian = 25
Integer jumlah_langkah_saat_ini = 0

Algoritma:
```
WHILE jumlah_langkah_saat_ini < batas_pendakian lakukan THEN
   tambah 1 ke jumlah_langkah_saat_ini
   PRINT "langkah ke: ", jumlah_langkah_saat_ini
   IF jumlah_langkah_saat_ini mod 5 = 0 maka
      PRINT "(menarik napas sejenak)"
   END IF
END WHILE
PRINT "istirahat. batas pendakian tercapai."
```

Flowchart:


Start
Integer batas_pendakian,
jumlah_langkah_saat_ini
batas_pendakian = 25
jumlah_langkah_saat_ini = 0
PRINT
"Elara memulai
pendakian"
WHILE
jumlah_langkah_saat_ini < batas_pendakian
True
jumlah_langkah_saat_init += 1
PRINT jumlah_langkah_saat_ini
IF
jumlah_langkah_saat_ini
% 5 == 0
False
True
PRINT umlah_langkah_saat_ini
False
PRINT batas_pendakian
Stop

Source Code:

```
batas_pendakian = 25
jumlah_langkah_saat_ini = 0

print("Elara memulai pendakian...")

while jumlah_langkah_saat_ini < batas_pendakian:
    jumlah_langkah_saat_ini += 1
    print(f"Langkah ke: {jumlah_langkah_saat_ini}")
    if jumlah_langkah_saat_ini % 5 == 0:
        print(f"(Menarik napas sejenak. Ini adalah langkah
ke-{jumlah_langkah_saat_ini})")
print(f"Istirahat. Batas pendakian {batas_pendakian} langkah tercapai.
Puncak sudah dekat.")
```

Output Program:

```
Elara memulai pendakian...
Langkah ke: 1
Langkah ke: 2
Langkah ke: 3
Langkah ke: 4
Langkah ke: 5
(Menarik napas sejenak. Ini adalah langkah ke-5)
Langkah ke: 6
Langkah ke: 7
Langkah ke: 8
Langkah ke: 9
Langkah ke: 10
(Menarik napas sejenak. Ini adalah langkah ke-10)
Langkah ke: 11
Langkah ke: 12
Langkah ke: 13
Langkah ke: 14
Langkah ke: 15
(Menarik napas sejenak. Ini adalah langkah ke-15)
Langkah ke: 16
Langkah ke: 17
Langkah ke: 18
Langkah ke: 19
Langkah ke: 20
(Menarik napas sejenak. Ini adalah langkah ke-20)
Langkah ke: 21
Langkah ke: 22
Langkah ke: 23
Langkah ke: 24
Langkah ke: 25
(Menarik napas sejenak. Ini adalah langkah ke-25)
Istirahat. Batas pendakian 25 langkah tercapai. Puncak sudah dekat.
```

Gambar 3.1 Output / Hasil Program

Bab 4: Menghitung Persediaan dengan Perulangan Terbatas

Angin di dataran tinggi bertiup kencang, menyibakkan rambut Elara dan membersihkan sisa-sisa kabut dari jubahnya. Di sini, di atap dunia, vegetasi menipis menjadi hamparan lumut dan bebatuan yang kokoh. Di belakangnya, Lembah Kelabu yang baru saja ia taklukkan tampak seperti lautan awan yang tenang. Pendakian yang melelahkan tadi telah menguras tenaganya, dan berat tas di punggungnya terasa berkali-kali lipat. Ini adalah momen yang tepat untuk berhenti, mengatur napas, dan yang terpenting: melakukan optimalisasi.

Dengan gerakan yang sudah menjadi ritual, Elara melepaskan tasnya dan meletakkannya di atas sebuah batu datar. Ia tidak akan membuang barang secara sembarangan. Setiap benda memiliki fungsi dan nilai, dan keputusannya harus didasarkan pada data, bukan sekadar perasaan lelah. Ia mengeluarkan Logika Logam dan meletakkannya di tengah, permukaannya yang dingin mulai berpendar lembut, siap untuk proses inventarisasi.

Satu per satu, ia mengeluarkan isi tasnya, seperti seorang pendeta yang menyiapkan persembahan. Ini adalah perulangan *for*-nya, sebuah proses yang akan ia jalankan untuk setiap item dalam "array" harta miliknya.

Pertama, "Obor Abadi". Cahayanya yang pucat tidak berkedip ditiup angin. Saat tangannya menyentuh logam dingin obor itu, Logika Logam di sebelahnya menampilkan bernilai 20. Obor ini adalah matanya di kegelapan, sebuah barang yang tak tergantikan.

Kedua, "Ransum". Lima bungkus tersisa. Praktis, vital, namun nilainya paling rendah. Logika Logam menilainya 5. Ini adalah bahan bakar untuk tubuhnya.

Ketiga, "Peta Usang". Ia membukanya dengan hati-hati, kertasnya yang rapuh menampilkan garis-garis yang sebagian sudah pudar. Peta ini lebih banyak memberikan pertanyaan daripada jawaban, namun menyimpan potensi. Nilainya 10.

Keempat, "Kompas Logika". Ia menyentuh artefak itu sendiri. Benda itu seakan mengenali dirinya, menampilkan nilai 15. Ini adalah otaknya, penuntunnya di tengah kekacauan.

Terakhir, ia mengangkat "Jubah Pelindung". Kainnya tebal, sedikit compang-camping di ujungnya, dan berbau asap dari api unggun yang tak terhitung jumlahnya. Jubah ini adalah kulit keduanya, pelindungnya dari dingin dan marabahaya. Logika Logam memberinya nilai tertinggi 30.

Ketika ia menyentuh jubah itu, perulangan berakhir. Semua item telah diiterasi. Di permukaan Logika Logam, glif-glif nilai individu memudar, lalu sebuah kalkulasi akhir muncul dengan cahaya yang terang: Total Nilai 80.

Elara menghela napas lega. Angka 80, menurut sistem nilai yang ia tetapkan, adalah ambang batas untuk "efisiensi maksimal". Setiap barang memiliki peran krusial membuang satu pun

akan lebih merugikan daripada keuntungan mengurangi sedikit beban. Berat di punggungnya adalah harga yang harus dibayar untuk kelangsungan hidup.

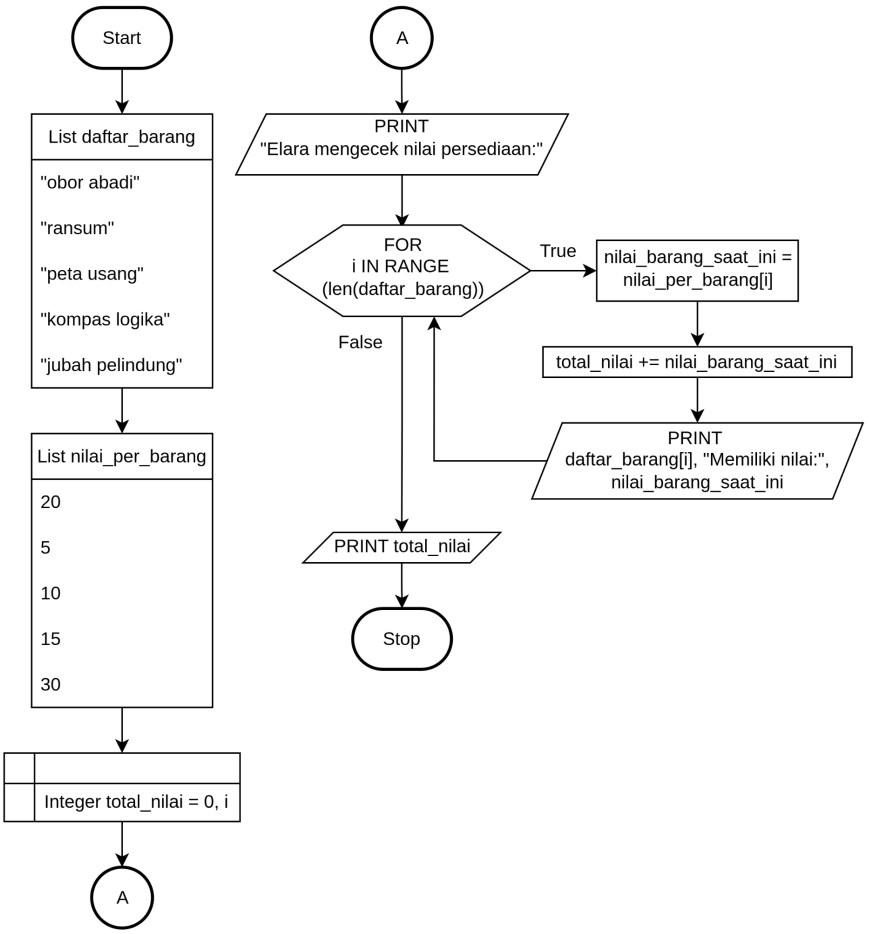
Dengan keyakinan baru, ia mengemas kembali setiap hartanya dengan rapi. Tasnya tetap terasa berat, tetapi keraguan di benaknya telah lenyap. Ia siap melanjutkan perjalanan.

Pseudocode :

Judul: Menghitung nilai total dari barang menggunakan perulangan FOR
Kamus / Deklarasi: List daftar_barang = ["obor abadi", "ransum", "peta usang", "kompas logika", "jubah pelindung"] Integer total_nilai = 0 List nilai_per_barang = [20, 5, 10, 15, 30]

Algoritma:
FOR setiap indeks i dari 0 hingga panjang(daftar_barang) - 1 THEN
  nilai_barang_saat_ini = nilai_per_barang[i]
  total_nilai += nilai_barang_saat_ini
  PRINT daftar_barang[i], " memiliki nilai: ", nilai_barang_saat_ini
END FOR
 	
PRINT "total nilai semua barang: ", total_nilai

Flowchart:



Source Code:

```
daftar_barang = ["Obor Abadi", "Ransum", "Peta Usang", "Kompas Logika", "Jubah
Pelindung"]
nilai_per_barang = [20, 5, 10, 15, 30]
total_nilai = 0

print("Elara mengecek nilai persediaan:")

for i in range(len(daftar_barang)):
    nilai_barang_saat_ini = nilai_per_barang[i]
    total_nilai += nilai_barang_saat_ini
    print(f"{daftar_barang[i]} memiliki nilai: {nilai_barang_saat_ini}")

print(f"\nTotal Nilai Semua Barang: {total_nilai} Koin Emas.")
```

Output Program:

```
Elara mengecek nilai persediaan:
Obor Abadi memiliki nilai: 20
Ransum memiliki nilai: 5
Peta Usang memiliki nilai: 10
Kompas Logika memiliki nilai: 15
Jubah Pelindung memiliki nilai: 30

Total Nilai Semua Barang: 80 Koin Emas.
```

Gambar 4.1 Output / Hasil Program

Bab 5: Pengaturan Inventaris yang Fleksibel

Perjalanan membawa Elara ke reruntuhan sebuah pos jaga tua, nyaris ditelan oleh cengkeraman akar-akar pohon raksasa. Di sudut yang terlindung dari cuaca, tergeletak sebuah peti kayu yang diperkuat besi. Rasa penasarannya tak terbendung. Dengan sedikit usaha, ia berhasil membuka tutupnya yang berat. Di dalamnya, di atas lapisan beludru yang telah usang, tergeletak dua benda yang tak ternilai.

Yang pertama adalah sebatang Kayu Sihir, terasa hangat saat disentuh dan memancarkan aroma pinus yang menenangkan benda ini bisa menyalakan api unggun hanya dengan satu perintah pikiran. Yang kedua adalah Batu Memori, sebuah kristal kuarsa yang berdenyut dengan cahaya redup, berisi informasi rute dari para pengelana terdahulu. Kegembiraan atas penemuannya segera diikuti oleh dilema logis tasnya sudah penuh.

Ia tidak bisa begitu saja memasukkan barang baru tanpa mengeluarkan yang lain. Ini adalah masalah manajemen inventaris. Ia membongkar isi tasnya di samping peti, menatap tumpukan ransumnya. Ia membawa tiga paket "Ransum Biasa" yang besar dan memakan tempat. Matanya lalu tertuju pada item terakhir di dalam peti satu lempengan tipis "Ransum Konsentrat". Logika Logam di sisinya berpendar, menampilkan simbol yang mengonfirmasi pikirannya satu ransum konsentrat memiliki nilai nutrisi setara dengan tiga ransum biasa, namun dengan ukuran sepertiganya.

Sebuah optimisasi cerdas langsung terbentuk di benaknya. Ia akan melakukan serangkaian operasi pada "list" inventarisnya. Pertama, menambahkan item baru. Kedua, menghapus item yang tidak efisien. Ketiga, menambahkan item pengganti yang lebih baik.

Dengan gerakan mantap, ia memasukkan Kayu Sihir dan Batu Memori ke dalam kantong khusus di tasnya sebuah operasi *append*. Kemudian, ia mengambil tiga paket Ransum Biasa yang besar dan meletakkannya di samping peti, mungkin akan berguna bagi pengelana lain tiga kali operasi *remove*. Terakhir, ia menyelipkan Ransum Konsentrat yang ramping ke tempat yang kosong satu lagi operasi *append*.

Tasnya kini terasa berbeda. Jumlah barangnya mungkin sama, tetapi kepadatannya terasa lebih efisien, lebih "dioptimalkan". Logika Logam di sakunya terasa sedikit lebih hangat, sebuah tanda kepuasan atas sistem yang berjalan dengan sempurna.

Pseudocode:

Judul: Manajemen Inventaris menggunakan List .append(), .remove(), len()
Kamus / Deklarasi: inventaris = ["Peta Usang", "Kompas Logika", "Ransum Biasa", "Ransum Biasa", "Ransum Biasa"]

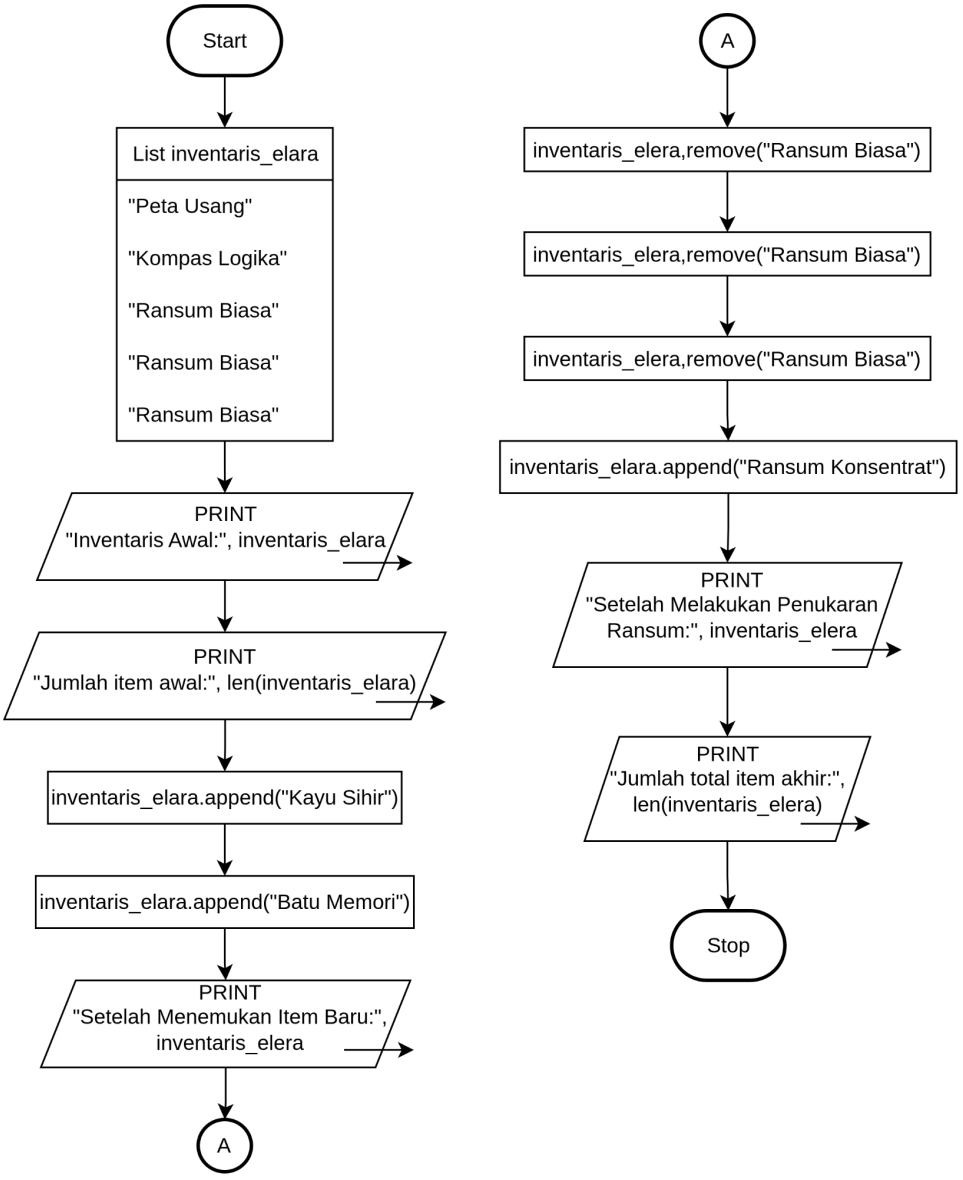
Algoritma:
tambahkan "kayu sihir" ke inventaris (append)
tambahkan "batu memori" ke inventaris (append)

hapus "ransum biasa" dari inventaris (remove)
hapus "ransum biasa" dari inventaris (remove)
hapus "ransum biasa" dari inventaris (remove)

tambahkan "ransum konsentrat" ke inventaris (append)

PRINT "isi inventaris akhir: ", inventaris
PRINT"jumlah total item: ", panjang(inventaris) (len)

Flowchart:



Source Code:

```
inventaris_elara = ["Peta Usang", "Kompas Logika", "Ransum Biasa", "Ransum
Biasa", "Ransum Biasa"]

print("Inventaris Awal:", inventaris_elara)
print("Jumlah item awal:", len(inventaris_elara))

inventaris_elara.append("Kayu Sihir")
inventaris_elara.append("Batu Memori")

print("\nSetelah Menemukan Item Baru:", inventaris_elara)

inventaris_elara.remove("Ransum Biasa")
inventaris_elara.remove("Ransum Biasa")
inventaris_elara.remove("Ransum Biasa")
inventaris_elara.append("Ransum Konsentrat")

print("\nSetelah Melakukan Penukaran Ransum:", inventaris_elara)
print("Jumlah total item akhir (len):", len(inventaris_elara))
```

Output Program:

```
Inventaris Awal: ['Peta Usang', 'Kompas Logika', 'Ransum Biasa', 'Ransum Biasa', 'Ransum Biasa']
Jumlah item awal: 5

Setelah Menemukan Item Baru: ['Peta Usang', 'Kompas Logika', 'Ransum Biasa', 'Ransum Biasa', 'Ransum Biasa', 'Kayu Sihir', 'Batu Memori']

Setelah Melakukan Penukaran Ransum: ['Peta Usang', 'Kompas Logika', 'Kayu Sihir', 'Batu Memori', 'Ransum Konsentrat']
Jumlah total item akhir (len): 5
```

Gambar 5.1 Output / Hasil Program

Bab 6: Koordinat Gerbang yang Tak Tergantikan

Perjalanan panjang Elara, yang dipandu oleh bisikan samar dari Batu Memori, akhirnya berakhir di sebuah ceruk tersembunyi di lereng gunung. Di hadapannya, menjulang sebuah Gerbang Eter Kuno. Struktur batu raksasa itu tampak menyatu dengan alam, ditutupi sulur-sulur tanaman merambat dan lumut, namun ukiran-ukiran rumit di permukaannya berdenyut dengan energi magis yang tak salah lagi. Ini adalah pintu menuju tempat lain, sebuah jalan pintas melalui jalinan dunia.

Elara mengangkat Batu Memori. Kristal itu bersinar terang, memproyeksikan serangkaian aksara cahaya ke udara di depannya. Informasi itu terbagi menjadi dua blok data yang berbeda. Blok pertama adalah serangkaian angka dan nama koordinat geografis gerbang itu. Blok kedua adalah kombinasi kata dan angka mantra aktivasi untuk tujuan spesifiknya.

Saat membaca data itu, Elara merasakan prinsip Logika Logam bekerja di benaknya. Data ini berbeda dari inventarisnya yang fleksibel. Ini adalah informasi absolut. Koordinatnya adalah fakta, kodenya adalah kunci. Kesalahan satu angka atau satu huruf saja tidak hanya akan membuat gerbang gagal berfungsi, tetapi bisa merobek struktur realitas atau membawanya ke kehampaan. Data seperti ini harus bersifat *immutable* tetap, abadi, dan tidak dapat diubah. Struktur data yang sempurna untuk ini adalah Tuple.

Dengan konsentrasi penuh, ia "mengunci" data pertama dalam pikirannya, membingkainya sebagai satu kesatuan yang tak terpisahkan: *(45.78, 120.33, 'North_Wing')*. Kemudian, ia melakukan hal yang sama untuk data kedua: *('Alpha', 7, 'Kuno')*. Keduanya kini menjadi blok data yang aman dan tidak bisa dimodifikasi secara tidak sengaja. Untuk mengaktifkan gerbang, ia hanya perlu menggabungkan kedua Tuple tersebut menjadi satu kunci utama.

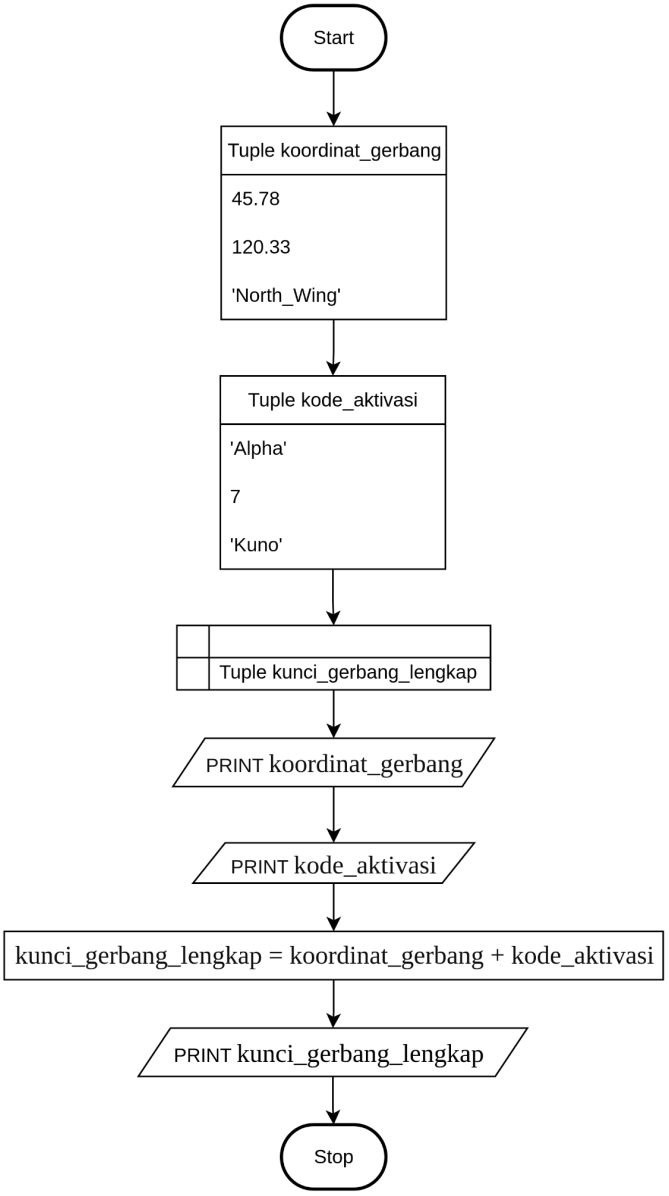
Dengan data yang benar dan *immutable*, Elara melangkah maju. Ia menyentuh ukiran di tengah gerbang, merasakan batu yang dingin bergetar di bawah telapak tangannya. Ia memproyeksikan "kunci lengkap" dari benaknya, membisikkannya bukan dengan suara, melainkan dengan pikiran. Seketika, ukiran di gerbang bersinar biru cemerlang, dan portal energi yang beriak seperti permukaan air terbuka di hadapannya. Ia melangkah masuk tanpa ragu.

Pseudocode:

Judul: Koordinat gerbang
Kamus / Deklarasi: Tuple koordinat_gerbang = (45.78, 120.33, 'North_Wing') Tuple kode_aktivasi = ('Alpha', 7, 'Kuno') Tuple kunci_gerbang_lengkap

Algoritma:
PRINT ("Koordinat Gerbang (Data Tetap):", koordinat_gerbang)
PRINT ("Kode Aktivasi (Data Tetap):", kode_aktivasi)
kunci_gerbang_lengkap = koordinat_gerbang + kode_aktivasi
PRINT ("Kunci Gerbang Lengkap (Hasil Gabungan):", kunci_gerbang_lengkap)

Flowchart:



Source Code:

```
koordinat_gerbang = (45.78, 120.33, 'North_Wing')
kode_aktivasi = ('Alpha', 7, 'Kuno')

print("Koordinat Gerbang (Data Tetap):", koordinat_gerbang)
print("Kode Aktivasi (Data Tetap):", kode_aktivasi)

kunci_gerbang_lengkap = koordinat_gerbang + kode_aktivasi

print("Kunci Gerbang Lengkap (Hasil Gabungan):", kunci_gerbang_lengkap)
```

Output Program:

```
Koordinat Gerbang (Data Tetap): (45.78, 120.33, 'North_Wing')
Kode Aktivasi (Data Tetap): ('Alpha', 7, 'Kuno')
Kunci Gerbang Lengkap (Hasil Gabungan): (45.78, 120.33, 'North_Wing', 'Alpha', 7, 'Kuno')
```

Gambar 6.1 Output / Hasil Program

Bab 7: Analisis Rune Unik

Gerbang Eter membawa Elara ke tempat yang menakjubkan Ruang Arsip Langit. Ruangan itu tidak memiliki dinding atau atap yang jelas, hanya hamparan lantai kristal yang melayang di tengah kosmos, dengan nebula dan bintang-bintang berkelip di kejauhan. Di atas lantai yang jernih itu, ratusan kepingan rune kuno tersebar seperti pecahan bintang jatuh. Masing-masing memancarkan cahaya elemen yang berbeda merah membara untuk api, biru beriak untuk air, hijau solid untuk tanah.

Menurut legenda yang ia dapat dari Batu Memori, setiap rune memiliki kekuatan elemen. Tugasnya adalah mengkatalog jenis-jenis kekuatan yang ada di ruangan itu. Ia mulai mengamati dan mencatat dalam benaknya "Rune Angin, Rune Tanah, Rune Angin lagi, Rune Air..." Ia segera menyadari masalahnya. Ada terlalu banyak duplikasi. Mencatat semuanya dalam sebuah daftar (*List*) hanya akan menghasilkan data mentah yang berantakan dan tidak efisien.

Ia butuh sebuah wadah konseptual yang secara otomatis menolak duplikat. Logika Logam di sisinya berdenyut lembut, seakan membisikkan solusi. Ia membayangkan sebuah karung ajaib. Setiap kali ia memasukkan sebuah rune ke dalamnya, karung itu akan memeriksa apakah rune sejenis sudah ada di dalam. Jika sudah ada, rune yang baru akan ditolak. Jika belum, rune itu akan diterima. Ini adalah prinsip kerja dari struktur data Set.

Dengan metode baru ini, pikirannya bekerja cepat. Ia "memindai" semua rune yang tersebar di lantai, melemparkannya secara mental ke dalam "karung himpunan" tersebut. "Rune Angin" masuk. "Rune Tanah" masuk. "Rune Angin" berikutnya? Ditolak. "Rune Air" masuk. "Rune Api" masuk. "Rune Api" berikutnya? Ditolak. Proses ini berjalan cepat hingga seluruh ruangan telah dipindai.

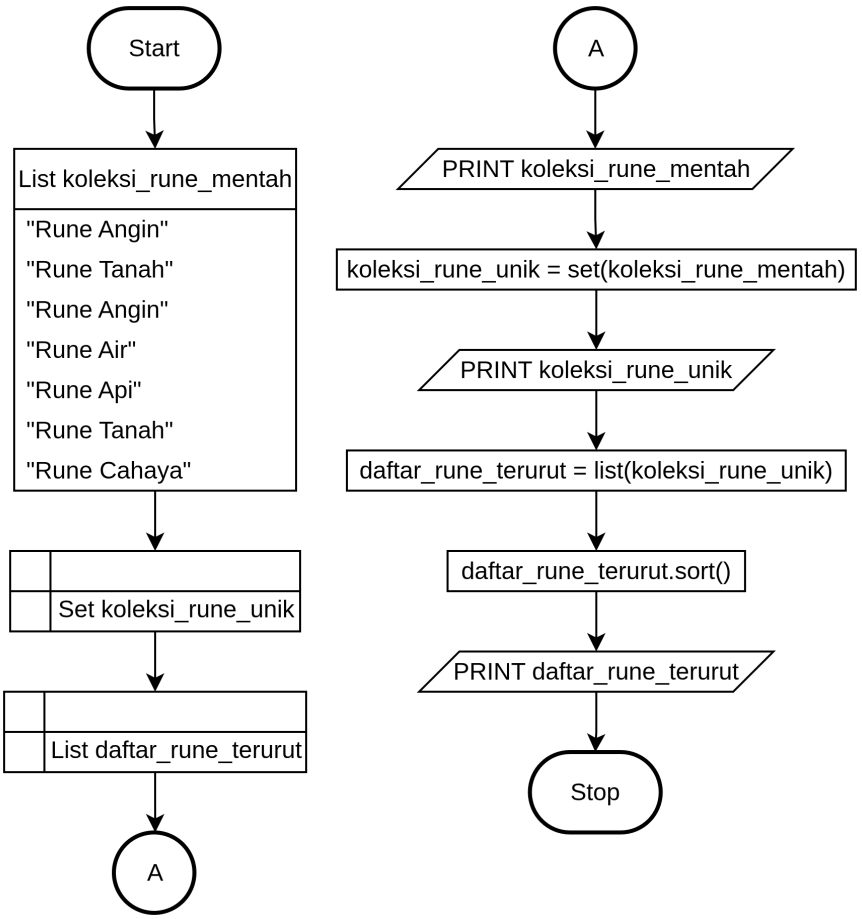
Hasilnya adalah koleksi rune yang unik, tetapi urutannya acak, persis seperti isi di dalam karung. Untuk membuatnya berguna, ia perlu langkah terakhir: pengurutan. Dengan satu perintah logika lagi, ia "menuangkan" isi karung itu ke dalam sebuah daftar baru dan menerapkan aturan *.sort()*. Di benaknya, kelima rune unik itu seketika mengatur diri mereka sendiri berdasarkan urutan abjad.

Elara tersenyum puas. Kekacauan ratusan rune kini telah disaring menjadi informasi yang jernih dan terstruktur. Ia sekarang tahu ia memiliki lima jenis rune. Lima elemen fundamental. Pengetahuan ini akan sangat berguna untuk perjalanan pulangnya.

Pseudocode:

Judul: Analisis Rune Unik
Kamus / Deklarasi: List koleksi_rune_mentah = ["Rune Angin", "Rune Tanah", "Rune Angin", "Rune Air", "Rune Api", "Rune Api", "Rune Tanah", "Rune Cahaya"] Set koleksi_rune_unik List daftar_rune_terurut
Algoritma: PRINT koleksi_rune_mentah koleksi_rune_unik = set(koleksi_rune_mentah) PRINT koleksi_rune_unik daftar_rune_terurut = list(koleksi_rune_unik) daftar_rune_terurut.sort() PRINT daftar_rune_terurut

Flowchart:



Source Code:

```
koleksi_rune_mentah = ["Rune Angin", "Rune Tanah", "Rune Angin", "Rune Air",
"Rune Api", "Rune Api", "Rune Tanah", "Rune Cahaya"]

print("Koleksi Rune Mentah (dengan Duplikasi):", koleksi_rune_mentah)

koleksi_rune_unik = set(koleksi_rune_mentah)
print("\nRune Unik yang Ditemukan (Menggunakan Set):", koleksi_rune_unik)

daftar_rune_terurut = list(koleksi_rune_unik)

daftar_rune_terurut.sort()
print("\nDaftar Rune Unik yang Telah Diurutkan (.sort()):", daftar_rune_terurut)
```

Output Program:

```
Koleksi Rune Mentah (dengan Duplikasi): ['Rune Angin', 'Rune Tanah', 'Rune Angin', 'Rune Air', 'Rune Api', 'Rune Api', 'Rune Tanah', 'Rune Cahaya']

Rune Unik yang Ditemukan (Menggunakan Set): {'Rune Cahaya', 'Rune Api', 'Rune Angin', 'Rune Tanah', 'Rune Air'}

Daftar Rune Unik yang Telah Diurutkan (.sort()): ['Rune Air', 'Rune Angin', 'Rune Api', 'Rune Cahaya', 'Rune Tanah']
```

Gambar 7.1 Output / Hasil Program

Bab 8: Perhitungan Strategi Perbekalan

Elara berdiri di ambang pintu keluar Ruang Arsip Langit, menatap jalur pulang yang terbentang di hadapannya sebuah medan berat yang menuntut ketahanan. Keberhasilan kini bukan lagi soal menemukan jalan, melainkan tentang perhitungan strategi yang cermat. Ia harus tahu pasti: berapa lama perbekalannya akan bertahan? Ia duduk bersila, mengeluarkan sisa Ransum Konsentrat miliknya. Lima paket dengan porsi yang sedikit berbeda itu ia anggap sebagai sebuah List data dalam benaknya, sebuah koleksi nilai *[2.0, 2.0, 2.0, 2.0, 1.0]*. Logika Logam di pangkuannya berpendar, siap menjadi mesin simulasi untuknya.

Langkah logis pertamanya adalah melakukan kalkulasi total. Pikirannya menjalankan sebuah For Loop secara mental, sebuah proses iterasi cepat yang menjumlahkan nilai dari setiap paket dalam daftarnya. Satu per satu, nilai-nilai itu diakumulasikan hingga Logika Logam menampilkan sebuah variabel krusial yang menjadi dasar perhitungannya *total_porsi* kini bernilai *9.0*. Dengan total energi yang sudah diketahui, ia memulai algoritma yang lebih kompleks, sebuah simulasi perjalanan dalam bentuk While Loop. Kondisi utamanya sederhana perulangan ini akan terus berjalan *selama total_porsi* yang ia miliki masih lebih besar atau sama dengan *konsumsi_harian* sebesar *1.5*.

Setiap putaran dalam perulangan ini merepresentasikan satu hari perjalanan. Di dalam siklus tersebut, penghitung *hari_bertahan* akan bertambah, dan *total_porsi* akan berkurang. Namun, Elara menambahkan sebuah variabel realisme ke dalam simulasinya, sebuah percabangan If-Else untuk memperhitungkan kelelahan. Jika penghitung hari menunjukkan angka ganjil, maka kelelahan ekstra akan menguras tambahan 0.2 porsi; jika tidak (pada hari genap), konsumsi akan tetap normal. Ia menyaksikan simulasi itu berjalan di permukaan artefaknya Hari ke-1, sisa porsi menjadi 7.3 setelah konsumsi normal dan ekstra. Hari ke-2, sisa porsi turun menjadi 5.8. Hari ke-3, sebagai hari ganjil, kembali menguras porsi ekstra, menyisakan 4.1. Angka itu terus menurun hingga pada akhir hari kelima, sisa porsi menunjukkan 0.9 jumlah yang tidak lagi mencukupi untuk memulai hari keenam.

Seketika, kondisi *While Loop* menjadi salah dan simulasi berhenti. Hasilnya terpampang jelas di permukaan artefak dan benaknya *5 hari penuh*. Informasi ini bukanlah kabar buruk, melainkan sebuah strategi yang solid. Tidak ada ruang untuk penundaan atau kesalahan. Dengan rencana yang matang dan rasa urgensi yang baru, Elara mengemasi perbekalannya dan memulai perjalanan pulangnya.

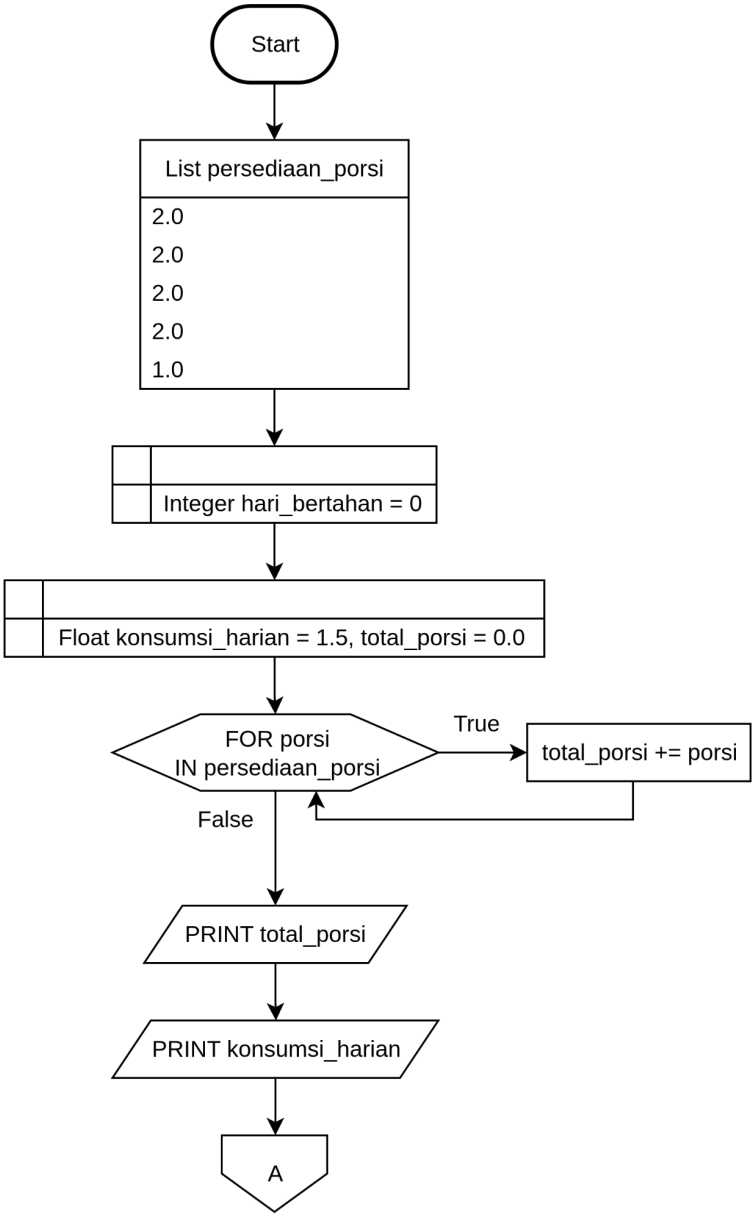
Pseudocode:

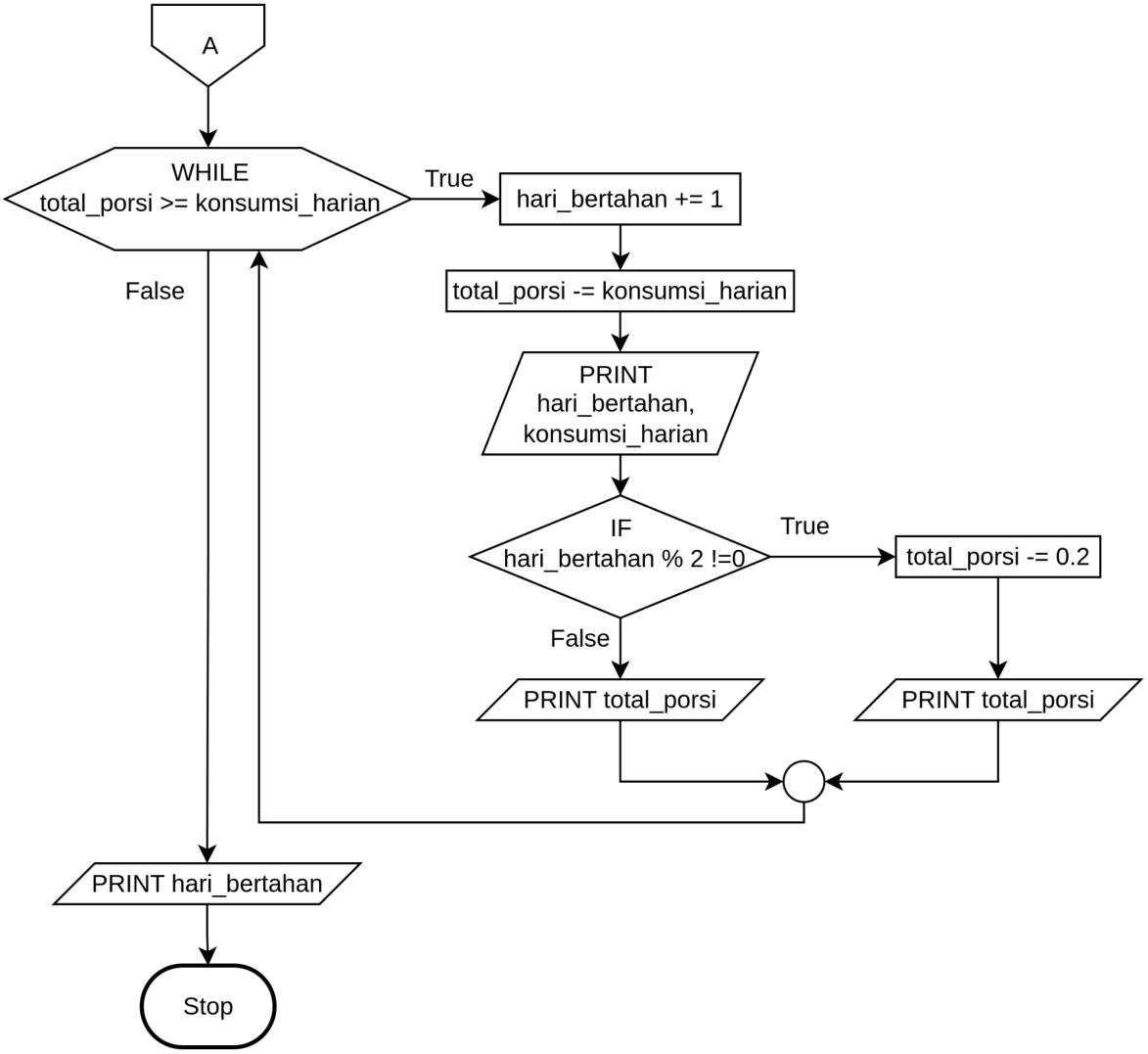
Judul:
Perhitungan Strategi Perbekalan

Kamus / Deklarasi:
List persediaan_porsi = [2.0, 2.0, 2.0, 2.0, 1.0]
Float konsumsi_harian = 1.5
Integer hari_bertahan = 0
Float total_porsi = 0.0

Algoritma:
```
FOR porsi IN persediaan_porsi
        total_porsi += porsi
PRINT total_porsi
PRINT konsumsi_harian
WHILE total_porsi >= konsumsi_harian
            hari_bertahan += 1
            total_porsi -= konsumsi_harian
            PRINT hari_bertahan, konsumsi_harian
            IF hari_bertahan % 2 != 0
               total_porsi -= 0.2
               PRINT total_porsi
            ELSE PRINT total_porsi
END WHILE PRINT hari_bertahan
```

Flowchart:


Start
List persediaan_porsi
2.0
2.0
2.0
2.0
1.0
Integer hari_bertahan = 0
Float konsumsi_harian = 1.5, total_porsi = 0.0
FOR porsi
IN persediaan_porsi
True
total_porsi += porsi
False
PRINT total_porsi
PRINT konsumsi_harian
A


A
WHILE
total_porsi >= konsumsi_harian
True
hari_bertahan += 1
total_porsi -= konsumsi_harian
PRINT
hari_bertahan,
konsumsi_harian
IF
hari_bertahan % 2 !=0
True
total_porsi -= 0.2
False
PRINT total_porsi
PRINT total_porsi
False
PRINT hari_bertahan
Stop

Source Code:

```
persediaan_porsi = [2.0, 2.0, 2.0, 2.0, 1.0]
konsumsi_harian = 1.5
hari_bertahan = 0
total_porsi = 0.0

for porsi in persediaan_porsi:
    total_porsi += porsi

print(f"Total Porsi Tersedia: {total_porsi}")
print(f"Konsumsi Harian: {konsumsi_harian} porsi\n" + "-"*20)

while total_porsi >= konsumsi_harian:
    hari_bertahan += 1
    total_porsi -= konsumsi_harian

    print(f"Hari ke-{hari_bertahan}: Konsumsi {konsumsi_harian} porsi.")
    if hari_bertahan % 2 != 0:
        total_porsi -= 0.2
        print(f"  -> Kelelahan (Hari Ganjil), konsumsi ekstra 0.2 porsi. Sisa:
{total_porsi:.1f}")
    else:
        print(f"  -> Sisa porsi: {total_porsi:.1f}")

print("-" * 20 + f"\nSIMULASI SELESAI. Elara dapat bertahan selama:
{hari_bertahan} hari penuh.")
```

Output Program:

```
Total Porsi Tersedia: 9.0
Konsumsi Harian: 1.5 porsi
--------------------
Hari ke-1: Konsumsi 1.5 porsi.
  -> Kelelahan (Hari Ganjil), konsumsi ekstra 0.2 porsi. Sisa: 7.3
Hari ke-2: Konsumsi 1.5 porsi.
  -> Sisa porsi: 5.8
Hari ke-3: Konsumsi 1.5 porsi.
  -> Kelelahan (Hari Ganjil), konsumsi ekstra 0.2 porsi. Sisa: 4.1
Hari ke-4: Konsumsi 1.5 porsi.
  -> Sisa porsi: 2.6
Hari ke-5: Konsumsi 1.5 porsi.
  -> Kelelahan (Hari Ganjil), konsumsi ekstra 0.2 porsi. Sisa: 0.9
--------------------
SIMULASI SELESAI. Elara dapat bertahan selama: 5 hari penuh.
```

Gambar 8.1 Output / Hasil Program

Bab 9: Pembentukan Tim dan Evaluasi Moral

Saat Elara bersiap untuk memulai perjalanan pulangnya yang penuh perhitungan, sebuah pemandangan tak terduga menyambutnya: kepulan asap tipis dari api unggun yang hampir padam. Di sekelilingnya, duduk tiga sosok yang tampak kelelahan. Ada Garen, seorang pria tegap berwajah keras yang hanya mengangguk saat Elara mendekat Lyra, seorang gadis yang matanya masih memancarkan keceriaan meskipun situasi mereka sulit; dan Zio, seorang pemuda yang duduk dengan bahu terkulai, tatapannya pesimis.

Setelah berbagi cerita singkat, mereka menyadari bahwa mereka semua tersesat dan memiliki tujuan yang sama. Dengan cepat, mereka sepakat untuk membentuk satu tim. Elara, dengan Logika Logam yang memberinya ketenangan dan kejelasan, secara alami mengambil peran sebagai navigator dan ahli strategi. Tugas pertamanya bukanlah menentukan arah, melainkan menganalisis aset terpentingnya sumber daya manusia. Ia perlu mengukur moral tim.

Pikirannya mulai bekerja, menerjemahkan interaksi sosial menjadi struktur data. Saat Garen berkeliling memeriksa perimeter dua kali, pikiran Elara yang menerapkan prinsip *Set* hanya mencatatnya satu kali, menghasilkan daftar anggota unik. Kemudian, ia mengamati dan menetapkan nilai moral dalam sebuah *List* Garen yang waspada mendapat nilai 8, Lyra yang optimis 9, dan Zio yang tampak putus asa hanya 7.

Langkah berikutnya adalah kalkulasi. Sebuah *For Loop* berjalan cepat di benaknya, menjumlahkan total moral: 8 + 9 + 7 = 24. Total itu kemudian ia bagi dengan jumlah anggota tim (3), menghasilkan *rata_rata_moral* sebesar 8.0. Angka ini menjadi standar kesehatan emosional tim mereka.

Dengan standar itu, ia menjalankan *For Loop* kedua, sebuah evaluasi individu dengan percabangan *If-Else*. Garen (8) memenuhi standar. Lyra (9) melebihinya. Namun, saat gilirannya Zio (7), kondisi *IF moral < rata_rata* terpenuhi. Sebuah "peringatan" mental muncul di benak Elara. Ini bukan sekadar angka; ini adalah data yang membutuhkan tindakan.

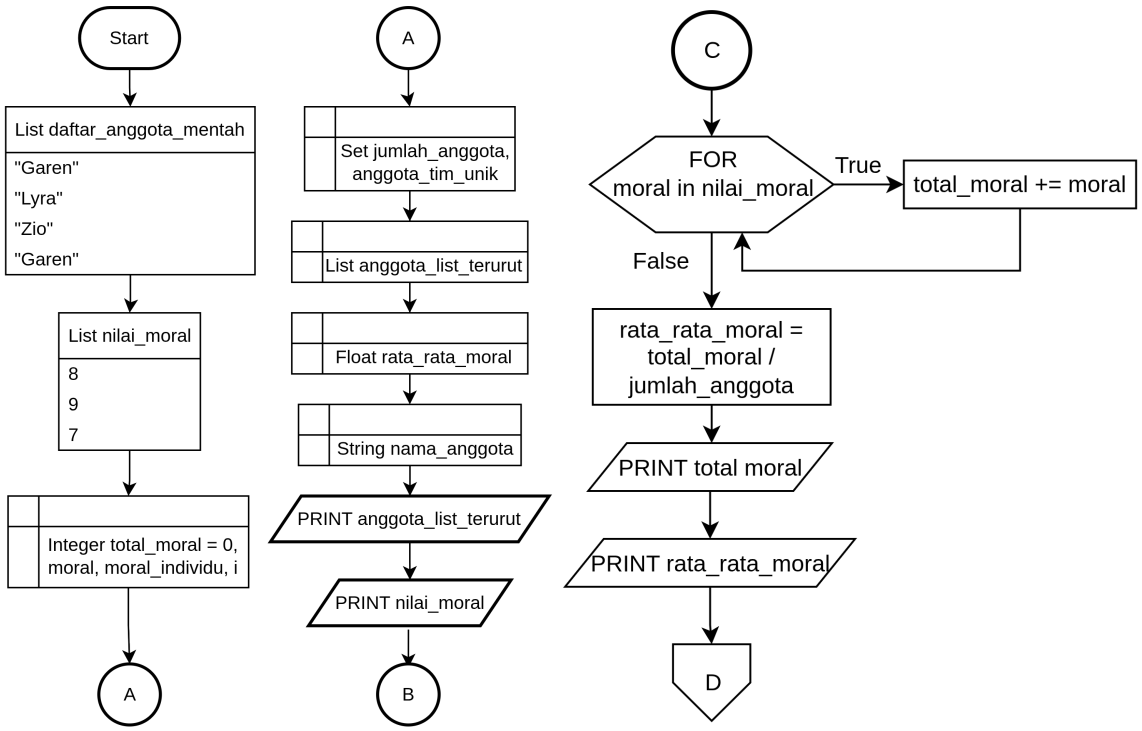
"Zio," kata Elara lembut sambil menghampirinya, memecah lamunannya. "Aku sudah menghitung, dan sepertinya semangatmu sedikit di bawah rata-rata tim kita. Makanlah Ransum Konsentrat ini, energimu akan pulih."

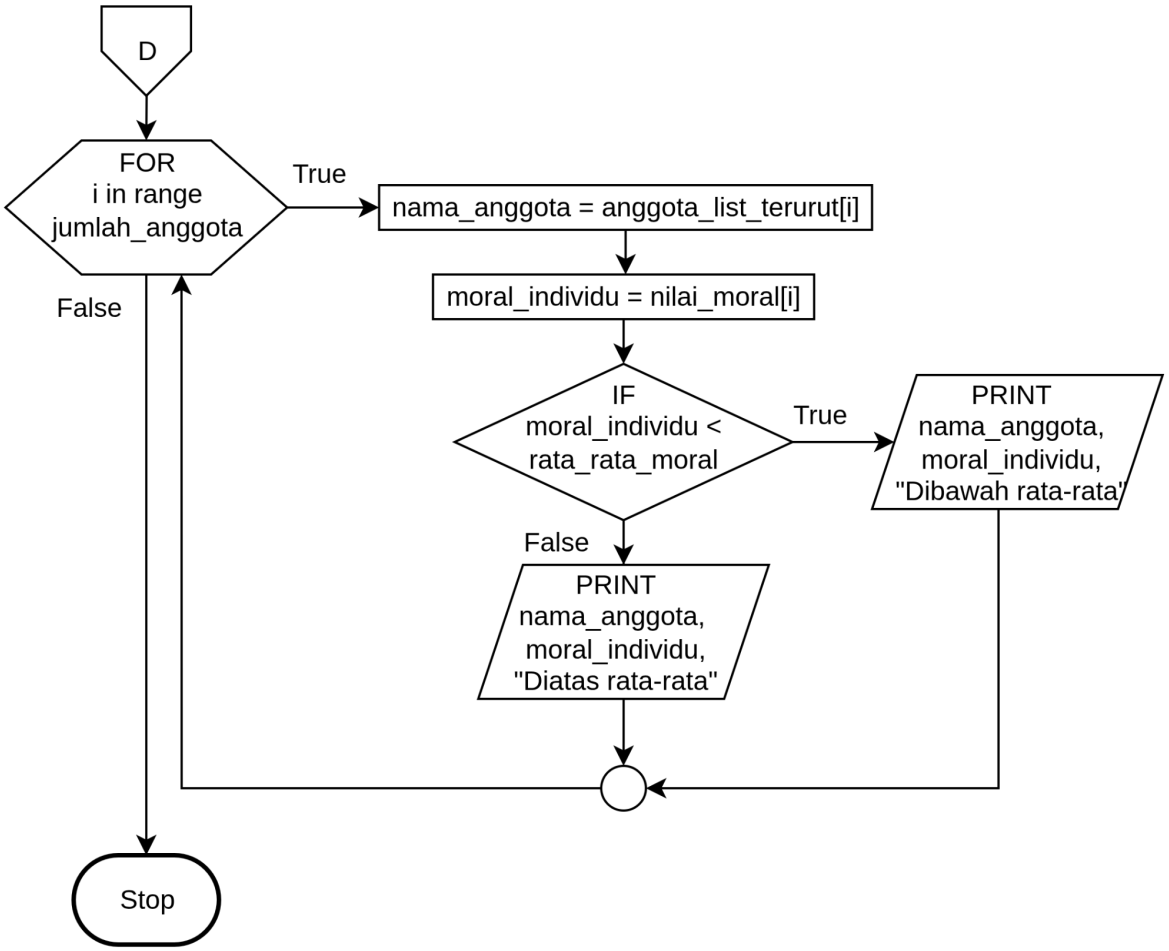
Zio terkejut, menatap Elara lalu ke ransum yang ditawarkan. Ia menerimanya dengan anggukan terima kasih yang tulus. Lyra tersenyum melihatnya, dan bahkan Garen menunjukkan sedikit senyum di sudut bibirnya. Logika tidak hanya membantu Elara bertahan hidup, tetapi juga membantunya berempati dan menjadi pemimpin yang baik.

Pseudocode:

Judul: Pembentukan tim dan evaluasi moral
Kamus / Deklarasi: List daftar_anggota_mentah = ["Garen", "Lyra", "Zio", "Garen"] List nilai_moral = [8, 9, 7] Integer total_moral = 0
Algoritma: anggota_tim_unik = set(daftar_anggota_mentah) jumlah_anggota = len(anggota_tim_unik) anggota_list_terurut = sorted(list(anggota_tim_unik)) PRINT anggota_list_terurut PRINT nilai_moral FOR moral in nilai_moral total_moral += moral rata_rata_moral = total_moral / jumlah_anggota END FOR PRINT total_moral PRINT rata_rata_moral FOR i IN RANGE jumlah_anggota nama_anggota = anggota_list_terurut[i] moral_individu = nilai_moral[i] IF moral_individu < rata_rata_moral PRINT nama_anggota, moral_individu, “Berada dibawah rata-rata” ELSE PRINT nama_anggota, moral_individu, “Berada di atas atau sama dengan rata-rata” END FOR

Flowchart:




D
FOR
i in range
jumlah_anggota
True
False
nama_anggota = anggota_list_terurut[i]
moral_individu = nilai_moral[i]
IF
moral_individu <
rata_rata_moral
True
False
PRINT
nama_anggota,
moral_individu,
"Dibawah rata-rata"
PRINT
nama_anggota,
moral_individu,
"Diatas rata-rata"
Stop

Source Code:

```
daftar_anggota_mentah = ["Garen", "Lyra", "Zio", "Garen"]
nilai_moral = [8, 9, 7]
total_moral = 0

anggota_tim_unik = set(daftar_anggota_mentah)
jumlah_anggota = len(anggota_tim_unik)
anggota_list_terurut = sorted(list(anggota_tim_unik))

print("Anggota Tim Unik Terurut:", anggota_list_terurut)
print("Nilai Moral (Garen, Lyra, Zio):", nilai_moral)
print("-" * 25)

for moral in nilai_moral:
    total_moral += moral
rata_rata_moral = total_moral / jumlah_anggota

print(f"Total Poin Moral: {total_moral}")
print(f"Rata-rata Moral Tim: {rata_rata_moral}")

print("\n--- Evaluasi Moral Individu ---")
for i in range(jumlah_anggota):
    nama_anggota = anggota_list_terurut[i]
    moral_individu = nilai_moral[i]

    if moral_individu < rata_rata_moral:
        print(f"Peringatan: {nama_anggota} (Moral {moral_individu}) berada di
bawah rata-rata. Beri motivasi!")
    else:
        print(f"Status: {nama_anggota} (Moral {moral_individu}) berada di atas
atau sama dengan rata-rata. Bagus!")
```

Output Program:

```
Anggota Tim Unik Terurut: ['Garen', 'Lyra', 'Zio']
Nilai Moral (Garen, Lyra, Zio): [8, 9, 7]
-------------------------
Total Poin Moral: 24
Rata-rata Moral Tim: 8.0

--- Evaluasi Moral Individu ---
Status: Garen (Moral 8) berada di atas atau sama dengan rata-rata. Bagus!
Status: Lyra (Moral 9) berada di atas atau sama dengan rata-rata. Bagus!
Peringatan: Zio (Moral 7) berada di bawah rata-rata. Beri motivasi!
```

Gambar 9.1 Output / Hasil Program

Bab 10: Jalan Terang Algoritma

Perjalanan pulang memakan waktu empat hari yang melelahkan. Namun, berkat perhitungan persediaan Elara (Bab 8) dan manajemen timnya (Bab 9), mereka tiba di perbatasan dengan selamat, hanya menyisakan 0.9 porsi makanan terakhir.

Sambil duduk di bawah sinar matahari pagi, Logika Logam di pangkuannya, Elara merenung. Pedang mungkin hebat untuk memotong monster, tetapi algoritma adalah senjata untuk memotong keraguan dan kekacauan.

"Setiap masalah besar," pikirnya, "adalah urutan dari langkah-langkah kecil. Aku hanya perlu melihatnya dengan benar."

Tipe Data Primitif memberinya pemahaman tentang variabel realitas. Ia belajar bahwa jarak (*Float*) berbeda dengan jumlah ransum (*Integer*), dan keduanya harus diperlakukan secara berbeda.

Percabangan (If-Else) memberinya kemampuan untuk memilih jalan terbaik, bukan berdasarkan emosi, tetapi berdasarkan data yang terukur—risiko versus sumber daya.

Perulangan (While dan For) adalah kekuatan untuk mengatasi tugas yang berulang. Baik itu mendaki bukit atau menghitung satu per satu item di tas, ia tahu bahwa dengan ketekunan logis, setiap pengulangan membawanya lebih dekat pada tujuan.

List mengajarkannya fleksibilitas—kemampuan untuk menambah (*append*) atau mengurangi (*remove*) beban, menyesuaikan diri dengan situasi baru. *Tuple* memberinya stabilitas—data penting seperti koordinat gerbang yang harus tetap suci. Dan *Set* membantunya menemukan esensi—hanya menyimpan item unik, membuang informasi yang tidak perlu.

Elara tersenyum. Ia bukan ksatria perkasa atau penyihir api, tetapi ia adalah seorang Petualang Algoritma. Ia telah mengubah jalan buntu menjadi serangkaian masalah yang dapat dipecahkan.

Maka, setiap petualang, entah di hutan belantara dunia fiksi atau di tengah tantangan kehidupan nyata, harus mengingat *Logika adalah Kompas Terbaik.*

Jangan biarkan masalah menjadi satu kesatuan yang menakutkan. Uraikan menjadi langkah-langkah, identifikasi datanya (*input*), definisikan kondisinya (*if-else*), ulangi proses yang dibutuhkan (*loop*), dan susunlah aset-aset Anda (*list/tuple/set*). Dari setiap jalan buntu yang dipecahkan, akan selalu ada Jalan Terang yang terbentang di hadapanmu. Logika akan selalu membawa kita pulang.