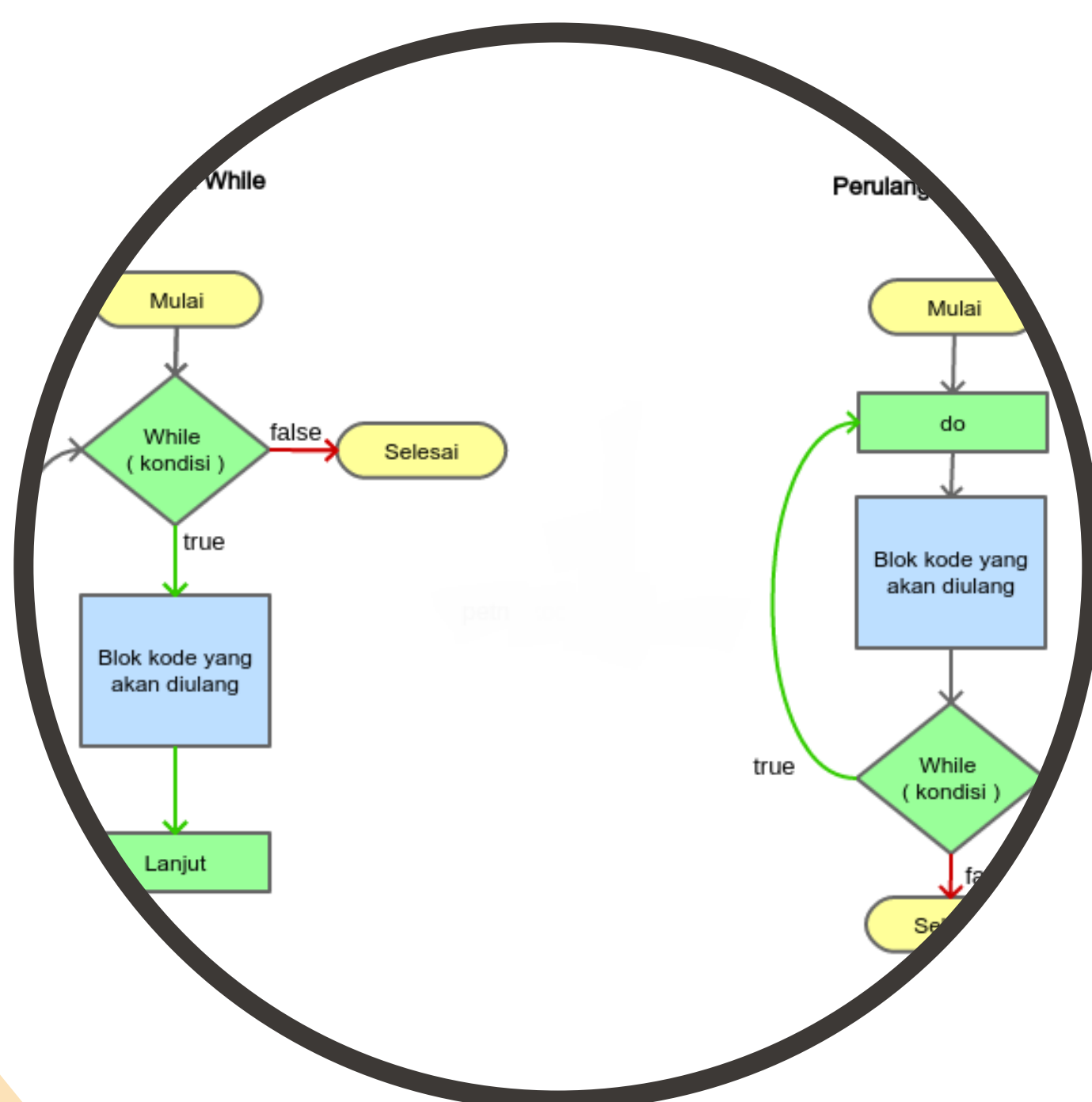


# Modul Percabangan Bagian I



**Disusun Oleh :**  
**Johannes Alexander Putra**

# Percabangan

Pernahkah Anda merasa ingin pergi ke suatu tempat menggunakan moda transportasi tertentu? Keputusan untuk memilih moda transportasi saat berpergian seringkali bergantung pada situasi tertentu. Sebagai contoh, ketika cuaca sedang hujan, biasanya orang lebih cenderung memilih mobil daripada sepeda motor. Namun, jika cuaca cerah dan jarak perjalanan tidak terlalu jauh, sepeda motor bisa menjadi pilihan yang lebih baik.



Gambar 1 Jika Hujan Mobil jadi Pilihan



Gambar 2 Jika Cerah Motor jadi Pilihan

Komputer, pada dasarnya, adalah alat yang membantu manusia dalam berbagai aktivitas. Ini bekerja dengan menjalankan perintah-perintah yang diberikan oleh manusia. Perintah-perintah ini diatur dalam bentuk kode program, yang berfungsi mengatur bagaimana komputer harus bertindak untuk menyelesaikan masalah tertentu. Proses ini juga melibatkan pengambilan keputusan, mirip dengan contoh pemilihan moda transportasi yang telah disebutkan sebelumnya. Dalam bagian ini, kami akan menjelaskan cara pengambilan keputusan dilakukan dalam program komputer. Istilah yang sering digunakan untuk ini adalah "kondisional." Apa itu kondisional? Secara sederhana, kondisional adalah cara untuk menyatakan "jika ..., maka ...". Ini digunakan untuk mengungkapkan tindakan yang harus diambil berdasarkan kondisi tertentu. Sebagai contoh, jika kita diminta untuk mengklasifikasikan apakah sebuah bilangan ganjil atau genap, kita dapat membuat aturan sebagai berikut (Musthofa, 2021):

1. Jika bilangan tersebut dapat dibagi habis oleh 2, maka bilangan tersebut diklasifikasikan sebagai bilangan genap.


$$2 \% 2 = 0?$$

**Gambar 3 Contoh Bilangan Genap**

2. Jika bilangan tersebut tidak dapat dibagi habis oleh 2, maka bilangan tersebut diklasifikasikan sebagai bilangan ganjil.

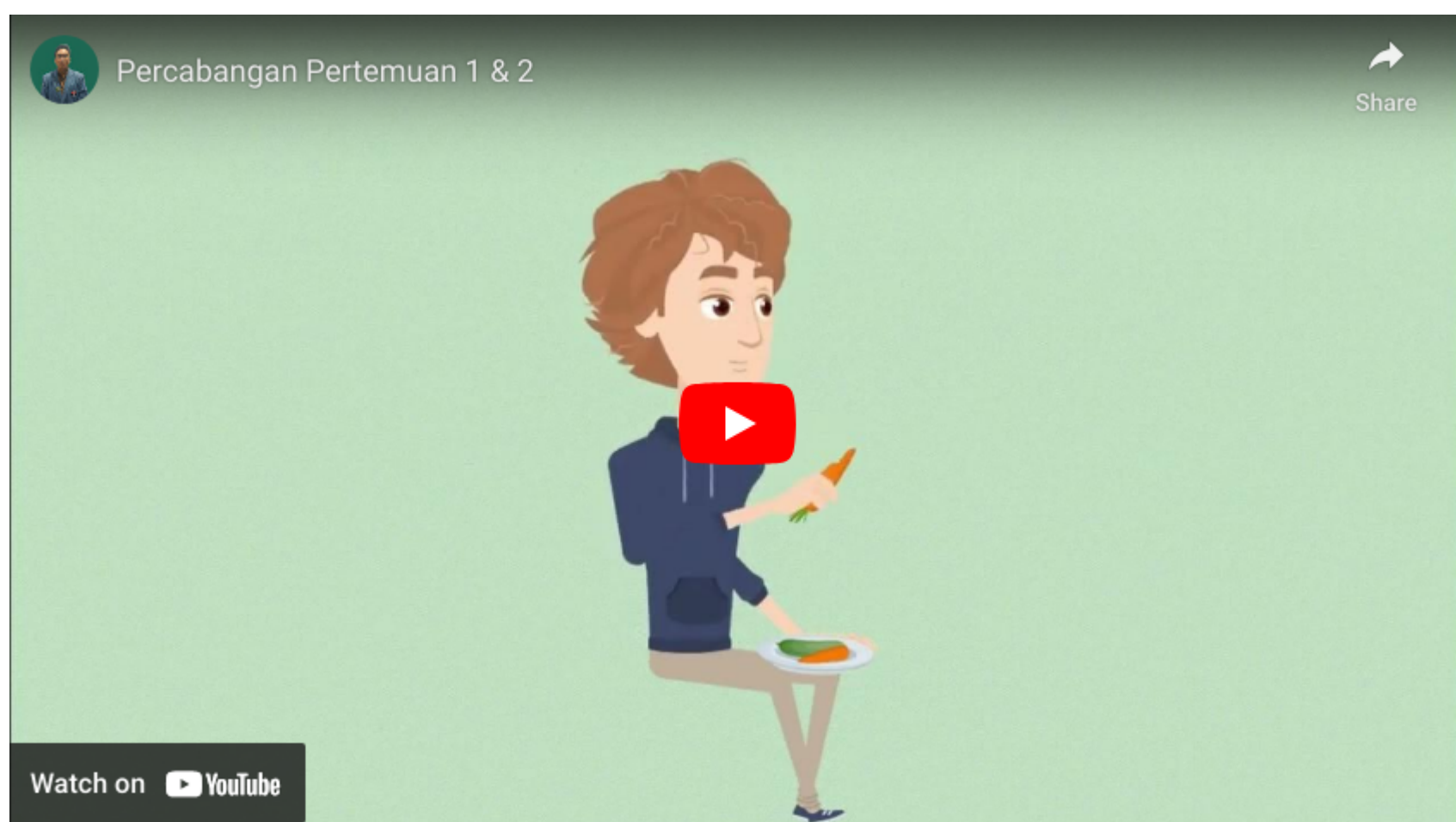

$$1 \% 2 = 0?$$

**Gambar 4 Contoh Bilangan Ganjil**

Terdapat beberapa jenis percabangan

- 1.Percabangan Tunggal (if)
- 2.Percabangan Dua Kasus (if else)
- 3.Percabangan Tiga Kasus atau lebih (if, else if, else)
- 4.Switch

Pada pertemuan pertama kita akan membahas mengenai percabangan tunggal dan mengimplementasikan dalam C++ dan Scratch.



<https://youtu.be/t4tAm87j5ic>

### Percabangan Tunggal

Percabangan Tunggal Blok program if untuk syarat satu kondisi berarti hanya ada sebuah blok aksi yang akan dikerjakan jika syarat kondisi terpenuhi. Jika kondisi yang ada pada jalannya program memenuhi syarat pada blok if, maka proses di dalam blok if akan dijalankan namun jika tidak memenuhi syarat blok if maka jalannya program tidak akan melakukan apa-apa (Munir & Lidya, 2016)

Contoh

Dalam suatu program akan menentukan apakah usia seseorang sudah manula, bila usia yang dimasukan adalah 60 atau lebih maka akan menampilkan output Manula. Jika tidak maka program tidak menampilkan apa-apa.

Berikut ini contoh-contoh Flowchart yang dapat digunakan



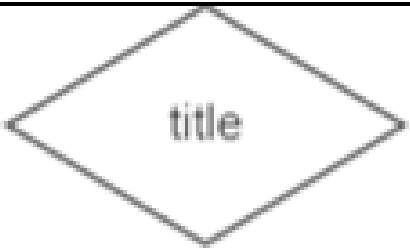
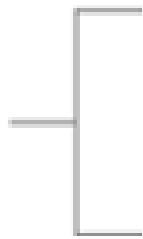



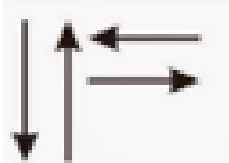



Tabel 1 Flowchart Percabangan 1 Kasus

| Flowchart Pertama   | Flowchart Kedua   | Flowchart Ketiga   |
|---|---|--|
| <pre>graph TD; A([mulai]) --&gt; B{{int usia}}; B --&gt; C[/masukan(usia)/]; C --&gt; D{usia &gt;=60}; D -- Ya --&gt; E[/cetak("Manula")/]; E --&gt; F([selesai]); D -- Tidak --&gt; F;</pre> | <pre>graph TD; A([mulai]) --&gt; B{{int usia}}; B --&gt; C[/usia/]; C --&gt; D{usia &gt;=60}; D -- Ya --&gt; E{{"Manula"}}; E --&gt; F([selesai]); D -- Tidak --&gt; F;</pre> | <pre>graph TD; A([Mulai]) --&gt; B[int usia]; B --&gt; C[/masukan(usia)/]; C --&gt; D{usia &gt;= 60}; D -- Ya --&gt; E[/cetak("Manula")/]; E --&gt; F([Selesai]); D -- Tidak --&gt; F;</pre> |

# Flowchart

Berikut ini merupakan tabel kegunaan simbol flowchart menurut Sukamto (2018) dan Setiawan (2017)

Tabel 2 Kegunaan Setiap Simbol Flowchart

| Simbol  | Nama                                       | Keterangan   |
|---|--|--|
|    | Proses                                     | Proses yang dilakukan secara internal di dalam komputer atau memori komputer   |
|   | Data                                       | Digunakan untuk beberapa operasi masukan/keluaran  |
|  | Keputusan                                  | Digunakan untuk pemilihan dalam bentuk dua jawaban (yes/no)  |
|  | Komentar                                   | Digunakan untuk menuliskan komentar dalam flowchart  |
|  | Inisialisasi                               | Digunakan untuk menggambarkan proses inisialisasi untuk blok perulangan for atau bisa juga untuk inisialisasi/deklarasi variable |
|  | Proses yang telah didefinisikan sebelumnya | Digunakan untuk memanggil sebuah rutin program atau bagian dari program (prosedur/fungsi)  |
|  | Pemberhentian                              | Digunakan untuk memulai atau mengakhiri proses   |
|  | Garis aliran                               | Digunakan untuk menunjuk arah aliran   |
|  | Menampilkan sesuatu ke layar               | Digunakan untuk menampilkan sesuatu ke layar   |
|  | Masukan manual                             | Digunakan jika ada masukan dari user   |
|  | Operasi manual                             | Biasanya jika ada blok perulangan yang dihentikan secara manual oleh user  |

Operator

Operator menurut Sukamto (2016) adalah simbol atau tanda yang jika diletakan pada dua buah operan dapat menghasilkan sebuah hasil, contohnya di dalam operasi matematika di mana tanda '+' diletakan diantara kedua buah nilai akan menghasilkan angka lain hasil penjumlahan kedua bilangam sebelumnya. Terdapat beberapa jenis operator

Tabel 3 Pembagian Operator



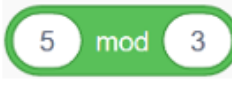


| Operator                | Keterangan   | Contoh                 |
|-------------------------|--|------------------------|
| Operator Aritmetik      | Operator yang biasa digunakan dalam matematika misalnya '+', '-', '/', '%', '*'  | 2 + 3                  |
| Operator Relasi         | Operator yang digunakan untuk membandingkan dua buah nilai. Misalnya ==, <, <=, >, >=, !=  | A<B                    |
| Operator Logika Boolean | Operator yang digunakan untuk mengaitkan dua buah ungkapan kondisi menjadi sebuah kondisi biasanya digunakan untuk melogikakan dua buah atau lebih kondisi contoh : && (AND) ,    (OR) | Kondisi_1 && Kondisi_2 |



Operator Aritmatika

Operator aritmatika merupakan operator yang palng sering digunakan karena pengolahan data sering berhubungan dengan nilai numerik (Jubilee Enterprise, 2017). Tabel dibawah ini menunjukan operator-operator aritmatika dalam bahasa pemrograman C++ dan Scratch.

Tabel 4 Operator Aritmatika

| Tanda | Nama                                      | C++   | Scratch   |
|-------|---|-------|---|
| *     | Operator perkalian                        | 2*3   |    |
| /     | Operator pembagian                        | 2/3   |  |
| %     | Operator sisa pembagian (Operator Modulo) | 5 % 3 |  |
| +     | Operator penjumlahan                      | 2 + 3 |  |
| -     | Operator pengurangan                      | 8 - 4 |  |

Operator Logika

Operator logika pada dasarnya membuat proses logika menjadi lebih kompleks. Dengan menggunakan operator ini pemrogram akan bisa menggabungkan beberapa kondisi. Contoh dari penggunaan operator ini: jika A dari Bandung DAN nilainya lebih tinggi dari 80, maka diberikan ucapan kelulusan menggunakan bahasa Sunda. Penggunaan kata 'dan' merupakan contoh dari operator logika (Jubilee Enterprise, 2017). Tabel di bawah ini menunjukan operator logika dalam bahasa C (Sukamto, 2018).



Tabel 5 Operator Logika

| Keterangan Operator   | Bahasa C++ | Scratch   |       |   |   |   |   |   |   |   |   |   |   |   |   |  |  |
|---|------------|-----------|-------|---|---|---|---|---|---|---|---|---|---|---|---|--|--|
| <p>Operator dan</p> <p>Operator logika yang menyatakan logika dan</p> <p>Keterangan:</p> <p>B : Benar</p> <p>S : Salah</p> <table><tr><th>Kondisi 1</th><th>Kondisi 2</th><th>Hasil</th></tr><tr><td>B</td><td>B</td><td>B</td></tr><tr><td>B</td><td>S</td><td>S</td></tr><tr><td>S</td><td>B</td><td>S</td></tr><tr><td>S</td><td>S</td><td>S</td></tr></table> | Kondisi 1  | Kondisi 2 | Hasil | B | B | B | B   | S | S | S | B | S | S | S | S | <p>Tanda: &amp;&amp;</p> <p>Contoh:</p> <p>(a&gt;b) &amp;&amp; (a !=b)</p> |  |
| Kondisi 1   | Kondisi 2  | Hasil     |       |   |   |   |   |   |   |   |   |   |   |   |   |  |  |
| B   | B          | B         |       |   |   |   |   |   |   |   |   |   |   |   |   |  |  |
| B   | S          | S         |       |   |   |   |   |   |   |   |   |   |   |   |   |  |  |
| S   | B          | S         |       |   |   |   |   |   |   |   |   |   |   |   |   |  |  |
| S   | S          | S         |       |   |   |   |   |   |   |   |   |   |   |   |   |  |  |
| <p>Operator atau</p> <p>Operator logika yang menyatakan logika atau</p> <table><tr><th>Kondisi 1</th><th>Kondisi 2</th><th>Hasil</th></tr><tr><td>B</td><td>B</td><td>B</td></tr><tr><td>B</td><td>S</td><td>B</td></tr><tr><td>S</td><td>B</td><td>B</td></tr><tr><td>S</td><td>S</td><td>S</td></tr></table>  | Kondisi 1  | Kondisi 2 | Hasil | B | B | B | B   | S | B | S | B | B | S | S | S | <p>Tanda:   </p> <p>Contoh:</p> <p>(a&gt;b)    (a !=b)</p>                 |  |
| Kondisi 1   | Kondisi 2  | Hasil     |       |   |   |   |   |   |   |   |   |   |   |   |   |  |  |
| B   | B          | B         |       |   |   |   |   |   |   |   |   |   |   |   |   |  |  |
| B   | S          | B         |       |   |   |   |   |   |   |   |   |   |   |   |   |  |  |
| S   | B          | B         |       |   |   |   |   |   |   |   |   |   |   |   |   |  |  |
| S   | S          | S         |       |   |   |   |   |   |   |   |   |   |   |   |   |  |  |
| <p>Operator Bukan</p> <p>Operator yang menyatakan logika bukan (negasi)</p> <table><tr><th>Kondisi 1</th><th>Hasil</th></tr><tr><td>B</td><td>S</td></tr><tr><td>S</td><td>B</td></tr></table>  | Kondisi 1  | Hasil     | B     | S | S | B | <p>Tanda: !</p> <p>Contoh:</p> <p>!(a==b)</p> |   |   |   |   |   |   |   |   |  |  |
| Kondisi 1   | Hasil      |           |       |   |   |   |   |   |   |   |   |   |   |   |   |  |  |
| B   | S          |           |       |   |   |   |   |   |   |   |   |   |   |   |   |  |  |
| S   | B          |           |       |   |   |   |   |   |   |   |   |   |   |   |   |  |  |

Operator Relasi

Operator relasi atau conditional operator adalah operator yang digunakan untuk membandingkan variable-variable yang umumnya bertipe data angka. Namun anda juga dapat menggunakannya untuk tipe data lainnya (Jubilee Enterprise, 2017). Table dibawah ini menunjukan operator relasi dalam bahasa C++ (Sukamto, 2018).

Tabel 6 Operator Relasi

| Keterangan Operator                   | Contoh dalam Bahasa C++ | Contoh dalam bahasa Scratch |
|---------------------------------------|-------------------------|-----------------------------|
| Operator sama dengan (=)              | a == b                  |                             |
| Operator tidak sama dengan            | a != b                  |                             |
| Operator lebih dari (>)               | a > b                   |                             |
| Operator kurang dari (<)              | a<b                     |                             |
| Operator lebih dari sama dengan (>=)  | a>=b                    |                             |
| Operator Kurang dari sama dengan (<=) | a<=b                    |                             |

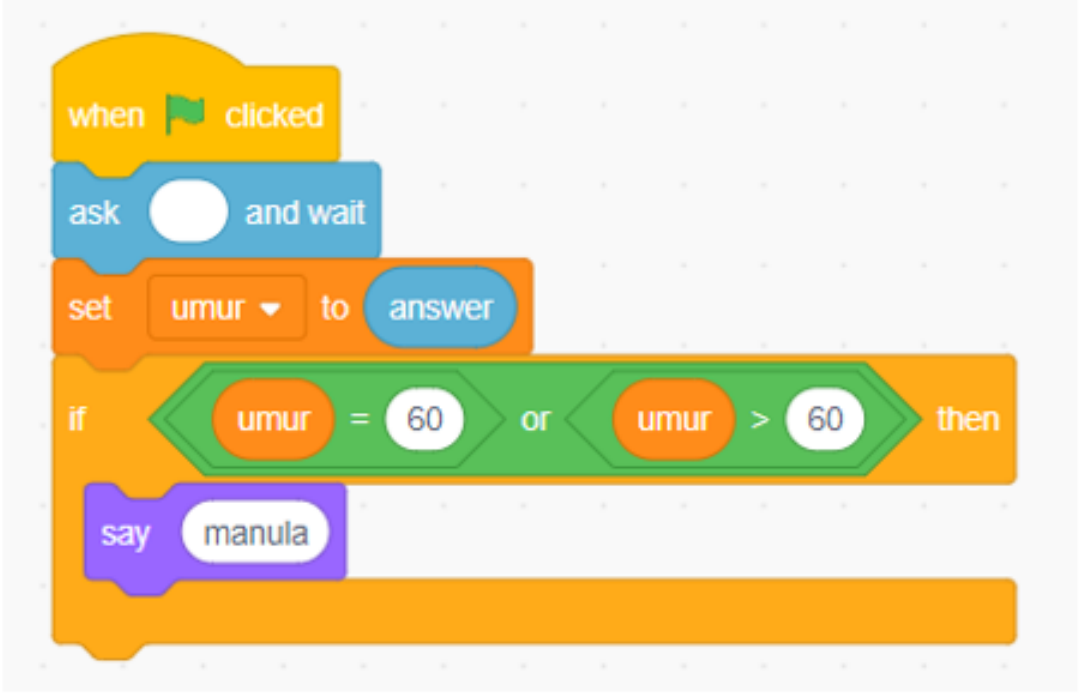
Struktur Percabangan Tunggal

Berikut ini merupakan struktur dari percabangan tunggal

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

Gambar 5 Struktur Percabangan Tunggal

Tabel 7 Perbandingan Kode Scratch dan C++

| Kode dalam Scratch  | Kode dalam C++   |
|---|--|
|  | <pre>#include &lt;iostream&gt; using namespace std;  int main() {     int umur;     cin&gt;&gt;umur;     if(umur &gt;= 60){         cout&lt;&lt;"Manula";     }      return 0; }</pre> |



[jdoodle.com/ia/Nh5](https://jdoodle.com/ia/Nh5)



<https://s.id/1Z397>

## Contoh Soal dan Penyelesaian

Seorang siswa ingin lulus ujian matematika. Siswa tersebut mengetahui bahwa untuk lulus ujian, siswa harus mendapatkan nilai minimal 70. Inputan nilai berasal dari user dan jika memenuhi syarat mengeluarkan output berupa “Lulus” dan tidak ada kondisi lainnya. Penamaan variable dibebaskan. Tipe data untuk nilai berupa integer. Program dibuat dalam bahasa pemrograman C++ dan Scratch

Menurut (Polya (2015) terdapat lima tahapan dalam pemecahan suatu permasalahan yaitu Memahami Masalah, Merencanakan Penyelesaian Masalah, Melaksanakan Rencana Penyelesaian Masalah, dan Memeriksa Kembali

### **Tahap 1 (Memahami Masalah):**

1 . Apakah saja data yang diketahui dan penting untuk menjawab permasalahan?

- Jika siswa ingin lulus ujian matematika harus mendapat nilai minimal 70.
- Jika siswa mendapatkan nilai minimal 70 maka mendapatkan output “Lulus”
- Bahasa pemrograman yang digunakan adalah C++ dan Scratch
- Inputan nilai berasal dari user
- Penamaan Variable dibebaskan

2. Apakah data yang diberikan sudah cukup untuk menyelesaikan pemecahan masalah? Sudah cukup, Data-data yang diberikan pada soal ini sudah cukup, Hal ini dikarenakan dengan data yang diberikan kita bisa mengetahui konsep apa yang dipergunakan dan membuat sebuah flowchart dan program

**Keterangan:**

**Kondisi dinyatakan berlebihan** bila data yang diberikan berlebihan contohnya, Jika terdapat soal: “Apakah kondisi tersebut cukup untuk menentukan bahwa percabangannya adalah percabangan tunggal?”, Jika pada soal diberikan 2 buah kondisi dan 2 buah aksi, tentu saja hal ini berlebihan.

**Kondisi dinyatakan kontradiktif.** Misalnya jika terdapat soal: “Apakah kondisi yang diberikan cukup untuk menjawab bahwa percabangan tersebut adalah percabangan tunggal?, Pada soal diberikan satu buah kondisi dan diberikan keterangan tidak ada kondisi dan aksi lainnya tetapi pada kalimat berikutnya diberikan aksi dan kondisi lainnya

**Kondisi dinyatakan cukup** bila data yang diberikan sudah cukup untuk menjawab pertanyaan. Contohnya terdapat soal: “Apakah kondisi tersebut cukup untuk menentukan bahwa percabangannya adalah percabangan tunggal?”, Pada soal diberikan 1 buah kondisi dan satu aksi

**Kondisi dinyatakan tidak cukup** bila data yang diberikan tidak cukup untuk menjawab pertanyaan. Contoh terdapat soal: “Apakah kondisi tersebut cukup untuk menentukan bahwa percabangannya adalah percabangan dua kasus?”, Pada soal diberikan sebuah kondisi saja. Tentu data yang diberikan kurang atau pada soal tersebut tidak diberikan aksi jika kondisi tercapai.

**Kondisi dinyatakan tidak ada konteks yang tepat** bila tidak menggambarkan sama sekali percabangan

3. Bila data yang diberikan ditambahkan, Jika siswa mendapatkan nilai kurang dari 70 maka mendapatkan output “Lulus”, Apakah data tersebut dapat menentukan bahwa percabangan yang digunakan adalah percabangan tunggal?

Data tersebut akan menjadikan soal Kontradiktif, karena pada soal diberikan data hanya memiliki 1 kondisi saja, kemudian dalam soal diberi tahu bahwa tidak kondisi lainnya, jadi ketika ditambahkan data: “Jika mendapat nilai dibawah 70” maka kondisi tersebut akan bertentangan dengan kondisi pada soal.



## **Tahap 2 (Membuat Rencana Penyelesaian Masalah):**

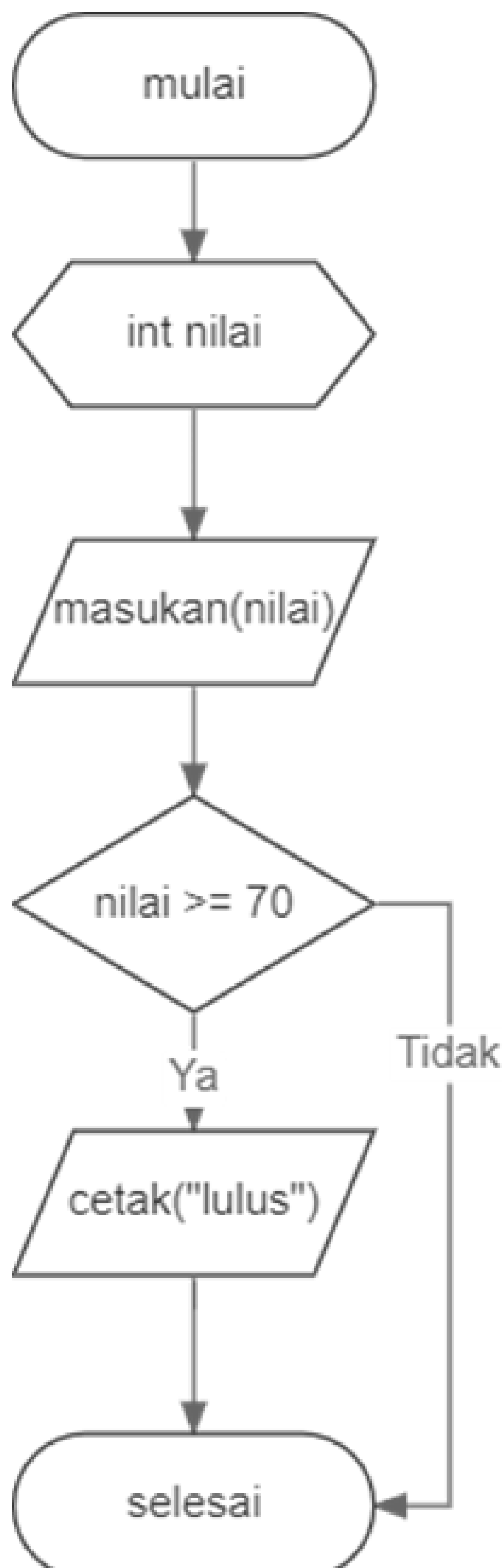
1. Kira-kira kondisi apakah yang mirip dengan kasus tersebut?

Jawab: kondisi yang mirip adalah percabangan tunggal, karena hanya ada satu aksi saja yang dilakukan

2. Bagaimana gambaran flowchart diagramnya?

Jawab:

Jika siswa tersebut memiliki nilai minimal 70 (70 atau lebih dari 70) maka siswa tersebut lulus ujian. Sedangkan jika tidak tidak dilakukan aksi apapun

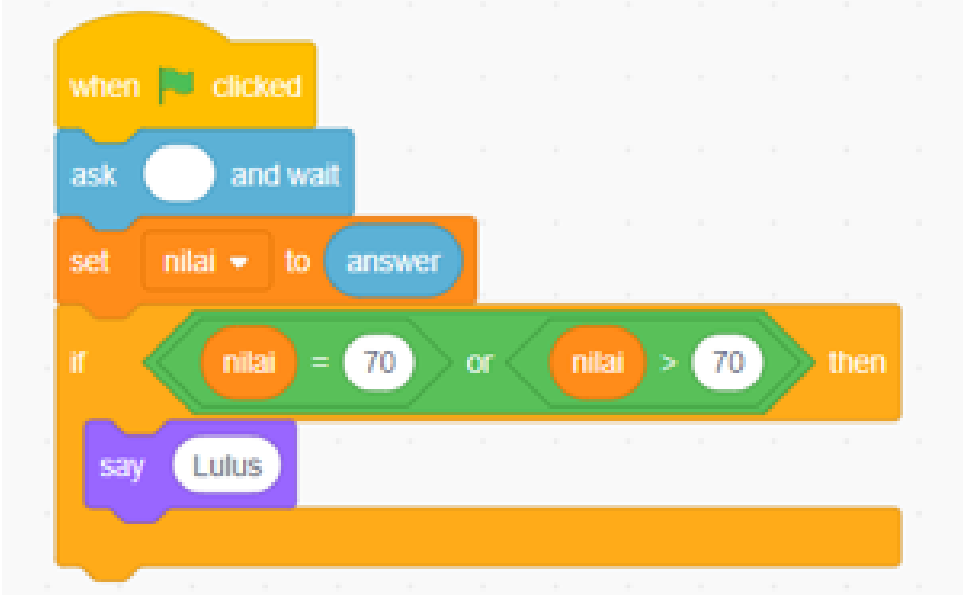




**Tahap 3 (Melaksanakan Rencana Penyelesaian Masalah):**

Bagaimana codenya untuk kasus tersebut?

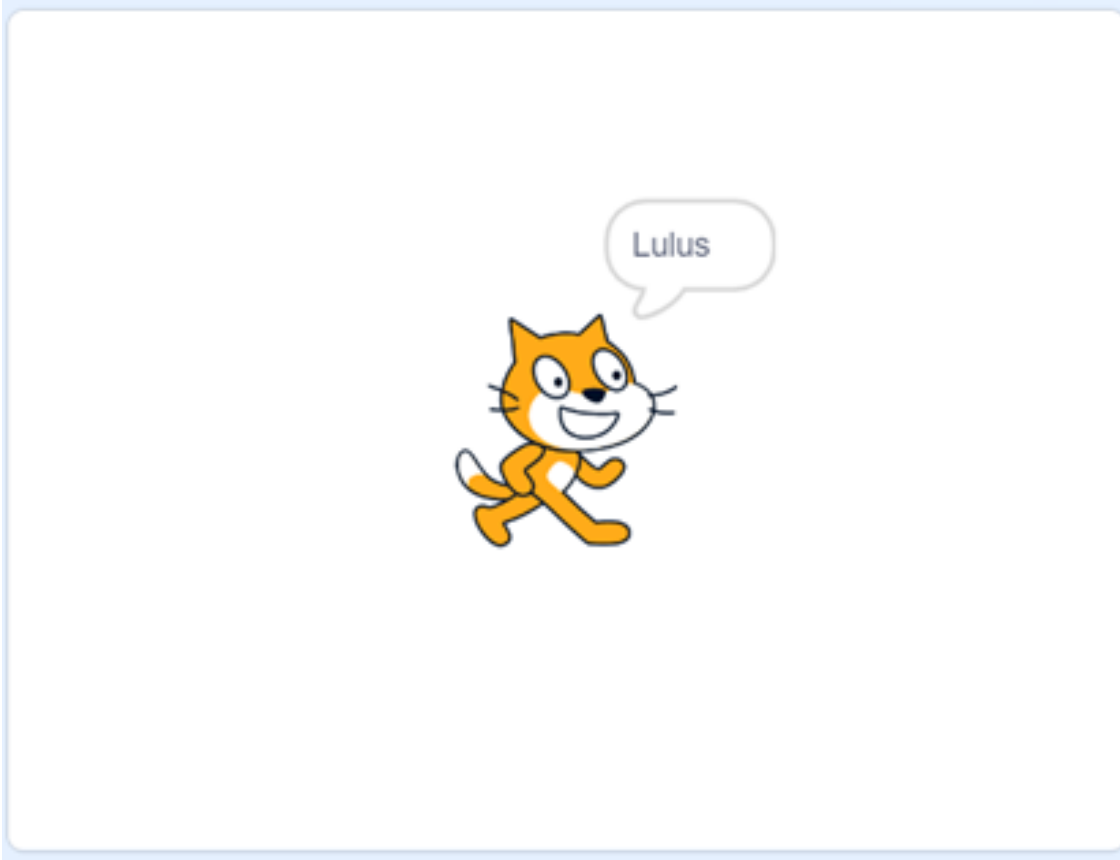
Jawab:

| Scratch  | C++  |
|--|--|
|  | <pre>1  #include &lt;iostream&gt; 2  using namespace std; 3  int main(){ 4      int nilai; 5      cin&gt;&gt; nilai; 6      if(nilai &gt;= 70){ 7          cout&lt;&lt;"Lulus"; 8      } 9      return 0; 10 }</pre> |

**Tahap 4 (Memeriksa Kembali)**

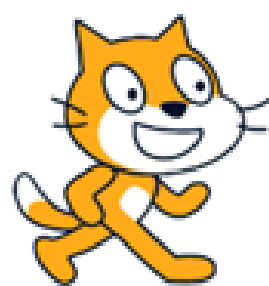
1. Apakah hasil pemecahan masalah sudah benar?

Ketika dimasukan 70 atau lebih maka menampilkan Lulus



Ketika dimasukan kurang dari 70. Program tidak melakukan apa-apa

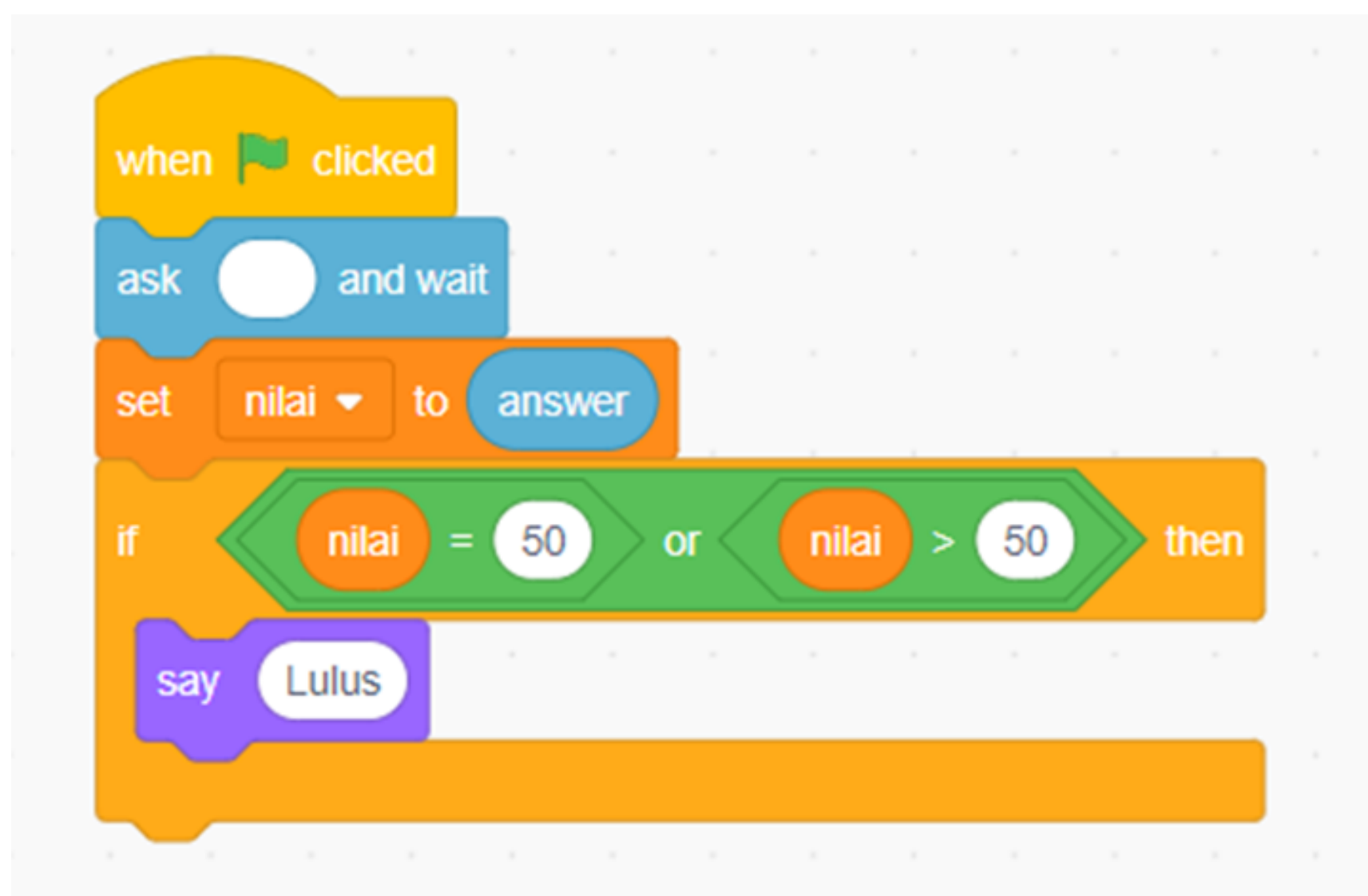
60



2. Bagaimana agar jika nilai minimal 50 maka menampilkan lulus?  
Mengubah kondisi dari `if(nilai >= 70)` menjadi `if(nilai >= 50)`

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4
5      int nilai;
6      cin>>nilai;
7      if(nilai >= 50){
8          cout<<"Lulus";
9      }
10
11      return 0;
12 }
```

Mengubah kondisi pada scratch yang sebelumnya `if nilai = 70 or nilai > 70` menjadi `if nilai = 50 or nilai > 50`



# Daftar Pustaka

Munir, R., & Lidya, L. (2016). Algoritma Dan Pemrograman Dalam Bahasa Pascal, C, C++ Edisi Keenam. Bandung: Informatika.

Polya, G. (2014). How to Solve It: A New Aspect of Mathematical Method. New Jersey: Princeton University Press.

Sukanto, R. A. (2018). Logika Algoritma dan Pemograman Dasar. Bandung: Modula.