

**MODUL**  
**PEMROGRAMAN DASAR**

---

**P E R U L A N G A N**

---

**2023**

## **A. KOMPETENSI DASAR**


3.7 Menerapkan struktur kontrol perulangan dalam bahasa pemrograman.

## **B. INDIKATOR PENCAPAIAN**

3.7.1 Menjelaskan statement/perintah untuk kontrol perulangan.

3.7.3 Menjelaskan statement/perintah untuk kontrol perulangan sederhana.

## **C. TUJUAN PEMBELAJARAN**

1. Siswa dapat menjelaskan statement/perintah untuk kontrol perulangan.
  2. Siswa dapat menjelaskan statement/perintah untuk kontrol perulangan sederhana.
- 

## **D. DASAR TEORI**

### **1. Definisi Perulangan**

Dalam bahasa pemrograman, tak terkecuali pada bahasa C++ terdapat suatu fasilitas atau fitur yang digunakan untuk melakukan proses yang berulang-ulang. Misalnya, ketika kita ingin mencetak tulisan "SMKN 2 Bandung" sebanyak 1 kali, mungkin kita tidak akan merasa kesulitan. Tapi bagaimana ketika kita harus mencetak kalimat "SMKN 2 Bandung" sebanyak 100 atau bahkan 1000 kali? Tentunya kita akan merasa kesulitan. Namun dengan fasilitas atau fitur perulangan pada bahasa pemrograman, kita tidak perlu menuliskan perintah sampai 100 atau 1000 kali, cukup dengan beberapa perintah saja.

Secara umum, perulangan dibagi menjadi dua kelompok yaitu counted loop dan uncounted loop.

- Counted loop, merupakan perulangan yang jelas dan sudah tentu banyak kali perulangannya. Contoh: Perulangan For
- Uncounted loop, merupakan perulangan yang tidak jelas berapa kali ia harus mengulang. Contoh: Perulangan While, Do-While

Struktur perulangan dalam bahasa C mempunyai bentuk yang bermacam-macam. Sebuah/kelompok instruksi diulang untuk jumlah pengulangan tertentu. Baik yang terdefinisikan sebelumnya atau tidak. Struktur pengulangan terdiri atas dua bagian:

- a. Kondisi pengulangan yaitu ekspresi boolean yang harus dipenuhi untuk melaksanakan pengulangan.
- b. Isi atau badan pengulangan yaitu satu atau lebih pernyataan (aksi) yang akan diulang.

## 2. Perulangan For

Perulangan for merupakan perulangan yang termasuk dalam counted loop, artinya perulangan ini biasa digunakan untuk mengulang suatu proses yang telah diketahui jumlah perulangannya. Dari segi penulisan, struktur perulangan for lebih efisien karena susunannya lebih simpel dan sederhana.

Bentuk penulisan perulangan for:

```
For (ekspresi1; ekspresi2; ekspresi3)
```

```
    Pernyataan 1;
```

```
    Pernyataan 2;
```

Penjelasan:

- Ekspresi 1: Digunakan untuk inisialisasi dari variabel yang dipakai untuk mengontrol ulangan eksekusi dari blok pernyataan yang ada di dalam pernyataan for (nilai awal dari pengulangan).
- Ekspresi 2: Digunakan untuk menyatakan suatu kondisi, eksekusi dari blok pernyataan yang ada di dalam pernyataan for akan diulang bila kondisi menghasilkan nilai **true**.
- Ekspresi 3: Digunakan untuk menambah/mengurangi nilai variabel yang dipakai untuk mengontrol eksekusi dari blok pernyataan yang ada di dalam pernyataan for

Untuk mempermudah dalam pembuatan perulangan for, langkah awal harus menentukan:

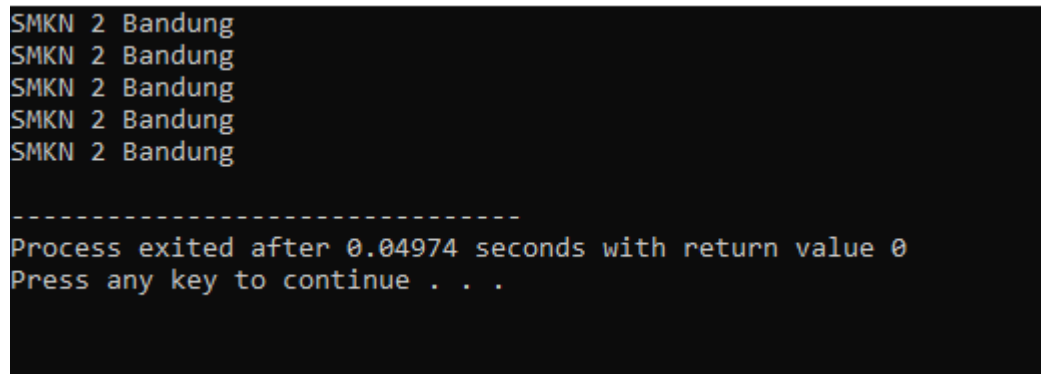
- Nilai awal dari pengulangan (nilai awal pengulangan tidak harus selalu dari 1)
- Nilai akhir pengulangan
- Nilai penaik/penurun dari pengulangan (counter)
- Blok pernyataan yang akan diulang
- Blok pernyataan setelah proses perulangan selesai

Contoh:

```
for(i=1; i<=5; i++)  
  
    cout<<"SMKN 2 Bandung"<<endl;
```

- Nilai awal pengulangan 1
- Nilai akhir pengulangan 5
- Nilai penarik adakah  $i++$  atau  $i = i + 1$
- Blok pernyataan yang diulang adalah `cout<<"SMKN 2 Bandung"<<endl`

Output:



```
SMKN 2 Bandung  
SMKN 2 Bandung  
SMKN 2 Bandung  
SMKN 2 Bandung  
SMKN 2 Bandung  
  
-----  
Process exited after 0.04974 seconds with return value 0  
Press any key to continue . . .
```

Penjelasan:

- Nilai awal pengulangan adalah 1 ( $i=1$ )
- Lakukan proses pengecekan, apakah nilai  $i$  ( $i=1$ ) masih memenuhi  $i<=5$ 
  - ya -> kerjakan bagian yang diulang (pernyataan 1) -> mencetak tulisan SMKN 2 Bandung
  - nilai  $i$  ditambah 1 ->  $i=2$
- Lakukan proses pengecekan, apakah nilai  $i$  ( $i=2$ ) masih memenuhi  $i<=5$ 
  - ya -> kerjakan bagian yang diulang (pernyataan 1) -> mencetak tulisan SMKN 2 Bandung
  - nilai  $i$  ditambah 1 ->  $i=3$
- Lakukan proses pengecekan, apakah nilai  $i$  ( $i=3$ ) masih memenuhi  $i<=5$ 
  - ya -> kerjakan bagian yang diulang (pernyataan 1) -> mencetak tulisan SMKN 2 Bandung

- nilai i ditambah 1 -> i=4
- Lakukan proses pengecekan, apakah nilai i (i=4) masih memenuhi i<=5
  - ya -> kerjakan bagian yang diulang (pernyataan 1) -> mencetak tulisan SMKN 2 Bandung
  - nilai i ditambah 1 -> i=5
- Lakukan proses pengecekan, apakah nilai i (i=5) masih memenuhi i<=5
  - ya -> kerjakan bagian yang diulang (pernyataan 1) -> mencetak tulisan SMKN 2 Bandung
  - nilai i ditambah 1 -> i=6
- lakukan proses pengecekan, apakah nilai i (i=6) masih memenuhi i<=5
  - tidak -> kerjakan pernyataan 2 dan baris seterusnya.

#### Studi Kasus

1.

```
int main(){
    int i;

    for(i=5; i>=1; i++){
        cout<<"SMKN 2 Bandung"<<endl;
    }
}
```

Output:

Program akan terus menerus mencetak pernyataan "SMKN 2 Bandung" karena nilai i akan selalu bertambah 1 tanpa batasnya karena pernyataan ekspresi 2  $i \geq 1$  terus memenuhi kondisi.

2.

```
int main(){
    int i;

    for(i=0; i>=10; i++){
        cout<<"SMKN 2 Bandung"<<endl;
    }
}
```

Output:

```
-----  
Process exited after 0.04688 seconds with return value 0  
Press any key to continue . . .
```

Program ketika dijalankan tidak akan mengeluarkan output apapun karena pernyataan ekspresi 1 tidak memenuhi persyaratan pernyataan ekspresi 2 ( $i = 0$ , 0 tidak  $\geq 10$ ).

3.

```
int main(){  
    int i;  
  
    for(i=1; i<=5; i++)  
        cout<<"a"<<endl;  
        cout<<"b"<<endl;  
        cout<<"c"<<endl;  
        cout<<"selesai";  
}
```

Output:

```
a  
a  
a  
a  
a  
b  
c  
selesai  
-----  
Process exited after 0.05062 seconds with return value 0  
Press any key to continue . . .
```

Program ketika dijalankan akan mengeluarkan output seperti diatas, hal ini dikarenakan program mendeteksi 4 pernyataan berbeda lalu mendeteksi perintah untuk melakukan pengulangan pernyataan "a" sebanyak 5 kali lalu berlanjut ke pernyataan berikutnya.

4.

```
int main(){
    int i;

    for(i=1; i<=5; i++){
        cout<<"a"<<endl;
        cout<<"b"<<endl;
        cout<<"c"<<endl;
    }
    cout<<"selesai";
}
```

Pernyataan 1

Pernyataan 2

Output:

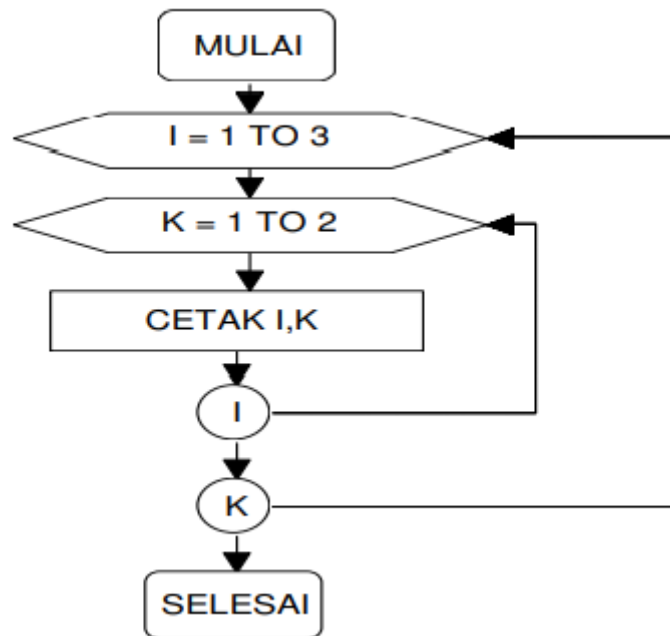
```
a
b
c
a
b
c
a
b
c
a
b
c
a
b
c
selesai
-----
Process exited after 0.04931 seconds with return value 0
Press any key to continue . . .
```

Program ketika dijalankan akan mengeluarkan output seperti di atas, hal ini dikarenakan program mendeteksi 3 pernyataan menjadi 1 pertanyaan utuh, sehingga program akan melakukan pengulangan pernyataan 1 hingga selesai lalu lanjut ke pernyataan 2. Untuk membuat agar beberapa perintah menjadi satu-kesatuan blok pernyataan gunakan tanda "{" sebagai awal blok dan tanda "}" sebagai akhir blok.



### 3. Perulangan For di dalam For (Kalang For)

Pada kasus tertentu, kadang diperlukan suatu proses yang berulang dan di dalam perulangan tersebut juga harus ada proses yang berulang.



Gambar diatas merupakan contoh flowchart kalang for. Proses dilakukan dengan mengulang nilai  $i=1$  dan saat proses ini dilakukan proses pengulangan  $k=1$  dan  $k=2$  dilakukan. Selanjutnya proses menambah nilai  $i=2$  dan dilakukan perulangan kembali  $k=1$  dan  $k=2$ . Kemudian nilai  $i$  bertambah menjadi  $i=3$ , proses perulangan  $k=1$  dan  $k=2$  juga berlangsung.

Contoh program:

```
int main(){
    int i, j;

    for(i=1; i<=3; i++){
        cout<<"A"<<endl;
        for (j=1; j<=5; j++){
            cout<<"B"<<endl;
        }
    }
    cout<<"Selesai";
}
```

Output:

```
A
B
B
B
B
B
B
A
B
B
B
B
B
A
B
B
B
B
B
Selesai
-----
Process exited after 0.05158 seconds with return value 0
Press any key to continue . . .
```

Penjelasan:

- For pertama akan melakukan proses pengulangan "A" selama 3 kali
- Bagian yang diulang juga proses pengulangan
  - Proses pengulangan 5 kali
- Urutan pengulangan
  - Ulangan i=1 : TRUE
    - Kerjakan pernyataan -> Cetak A
      - Ulangan j=1 : TRUE
        - Kerjakan pernyataan -> Cetak B
      - Ulangan j=2 : TRUE
        - Kerjakan pernyataan -> Cetak B
      - Ulangan j=3 : TRUE
        - Kerjakan pernyataan -> Cetak B
      - Ulangan j=4 : TRUE
        - Kerjakan pernyataan -> Cetak B
      - Ulangan j=5 : TRUE
        - Kerjakan pernyataan -> Cetak B
      - Ulangan j=6 : FALSE

- Kembali ke pengulangan i
- Ulangan i=2 : TRUE
  - Kerjakan pernyataan -> Cetak A
    - Ulangan j=1 : TRUE
      - Kerjakan pernyataan -> Cetak B
    - Ulangan j=2 : TRUE
      - Kerjakan pernyataan -> Cetak B
    - Ulangan j=3 : TRUE
      - Kerjakan pernyataan -> Cetak B
    - Ulangan j=4 : TRUE
      - Kerjakan pernyataan -> Cetak B
    - Ulangan j=5 : TRUE
      - Kerjakan pernyataan -> Cetak B
    - Ulangan j=6 : FALSE
      - Kembali ke pengulangan i
- Ulangan i=3 : TRUE
  - Kerjakan pernyataan -> Cetak A
    - Ulangan j=1 : TRUE
      - Kerjakan pernyataan -> Cetak B
    - Ulangan j=2 : TRUE
      - Kerjakan pernyataan -> Cetak B
    - Ulangan j=3 : TRUE
      - Kerjakan pernyataan -> Cetak B
    - Ulangan j=4 : TRUE
      - Kerjakan pernyataan -> Cetak B
    - Ulangan j=5 : TRUE
      - Kerjakan pernyataan -> Cetak B
    - Ulangan j=6 : FALSE
      - Kembali ke pengulangan i
- Ulangan i=4 : FALSE
  - Kerjakan pernyataan 2 -> Cetak selesai

#### 4. Perulangan While

Pernyataan ini digunakan untuk mengulang eksekusi dari suatu blok pernyataan yang jumlah ulangnya tergantung dari kondisi yang diberikan, sejauh kondisi TRUE, maka pengulangan dari blok tersebut akan terus dilakukan. Format pernyataan While adalah

```
While (kondisi){  
    Blok pernyataan 1  
    Blok control  
}  
Blok pernyataan 2
```

##### Contoh:

```
int i;  
i = 1;  
while(i <= 5)  
    cout<<"SMKN 2 Bandung"<<endl;  
    i++;
```

- Nilai awal i = 1
- Nilai akhir i = 5
- Nilai penaik adakah i++ atau i = i + 1
- Blok pernyataan yang diulang adalah cout<<"SMKN 2 Bandung"<<endl

##### Output:

```
SMKN 2 Bandung
SMKN 2 Bandung
SMKN 2 Bandung
SMKN 2 Bandung
SMKN 2 Bandung

-----
Process exited after 0.04974 seconds with return value 0
Press any key to continue . . .
```

### Penjelasan:

- Nilai awal perulangan adalah 1 ( $i=1$ )
- Lakukan proses pengecekan, apakah nilai  $i$  ( $i=1$ ) masih memenuhi  $i \leq 5$ 
  - ya -> kerjakan bagian yang diulang (pernyataan 1) -> mencetak tulisan SMKN 2 Bandung
  - nilai  $i$  ditambah 1 ->  $i=2$
- Lakukan proses pengecekan, apakah nilai  $i$  ( $i=2$ ) masih memenuhi  $i \leq 5$ 
  - ya -> kerjakan bagian yang diulang (pernyataan 1) -> mencetak tulisan SMKN 2 Bandung
  - nilai  $i$  ditambah 1 ->  $i=3$
- Lakukan proses pengecekan, apakah nilai  $i$  ( $i=3$ ) masih memenuhi  $i \leq 5$ 
  - ya -> kerjakan bagian yang diulang (pernyataan 1) -> mencetak tulisan SMKN 2 Bandung
  - nilai  $i$  ditambah 1 ->  $i=4$
- Lakukan proses pengecekan, apakah nilai  $i$  ( $i=4$ ) masih memenuhi  $i \leq 5$ 
  - ya -> kerjakan bagian yang diulang (pernyataan 1) -> mencetak tulisan SMKN 2 Bandung
  - nilai  $i$  ditambah 1 ->  $i=5$
- Lakukan proses pengecekan, apakah nilai  $i$  ( $i=5$ ) masih memenuhi  $i \leq 5$ 
  - ya -> kerjakan bagian yang diulang (pernyataan 1) -> mencetak tulisan SMKN 2 Bandung
  - nilai  $i$  ditambah 1 ->  $i=6$

- lakukan proses pengecekan, apakah nilai i (i=6) masih memenuhi  $i \leq 5$ 
  - tidak -> program keluar dari pengulangan dan kerjakan pernyataan 2 dan baris seterusnya.

Kondisi dalam While juga dapat menggunakan operator logika. Misalnya:

While (kondisi 1) && (kondisi 2){

Blok pernyataan 1

Blok control

}

Blok pernyataan 2

### Perbedaan While dengan For:

#### Program While

```
int main(){
int i;
i = 1;
while (i <= 5){
    cout<<"SMKN 2 Bandung"<<endl;
    i++;
}
cout<<"Selesai";
}
```

#### Program For

```
int main(){
int i;
for(i=1; i<=5; i++){
    cout<<"SMKN 2 Bandung"<<endl;
}
cout<<"Selesai";
}
```

### Contoh program:

1)

```
int main(){
    int awal;
    awal = 5;
    while (awal <= 3){
        awal = awal + 1;
        cout<<"SMKN 2 Bandung"<<endl;
    }
}
```

Penjelasan:

- Program di atas jika dijalankan tidak ada hasilnya, hal ini karena nilai awal=5

- Kemudian dilakukan pengecekan apakah awal  $\leq 3 \rightarrow 5 \leq 3$ 
  - Salah, sehingga tidak terjadi proses pengulangan

2)

```
int main(){
    int awal;
    awal = 1;
    while (awal <= 3){
        //awal = awal + 1;
        cout<<"SMKN 2 Bandung"<<endl;
    }
}
```

Penjelasan:

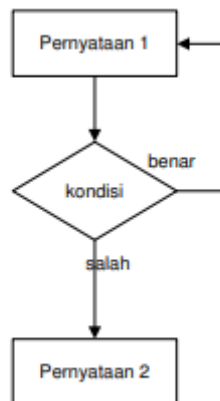
- Program di atas jika dijalankan akan berulang terus menerus (*infinite loop*), hal ini karena nilai awal yang berfungsi sebagai control (pencacah) tidak bertambah sehingga nilai akan selalu benar.

## 5. Perulangan Do – While

Perintah ini dipakai untuk mengulang eksekusi dari suatu blok pernyataan yang jumlah ulangnya tergantung dari kondisi yang diberikan, selama kondisinya TRUE, maka pengulangan akan terus dilakukan. Secara umum, perintah ini hampir sama dengan perintah WHILE. Format penulisannya:

```
Do
{
    Blok pernyataan;
}
While (kondisi);
Blok pernyataan 2;
```

Pada Do While, blok pernyataan akan dikerjakan terlebih dahulu baru memeriksa hasil kondisi. Jika kondisi bernilai TRUE, maka blok pernyataan akan dikerjakan lagi (terjadi proses pengulangan) dan bila kondisi FALSE, program akan keluar dari proses pengulangan.



### Contoh:

```
int i;
i = 1;
do
    cout<<"SMKN 2 Bandung"<<endl;
```



```
i++;  
while(i <= 5)
```

- Nilai awal i = 1
- Nilai akhir i = 5
- Nilai penaik adakah i++ atau i = i + 1
- Blok pernyataan yang diulang adalah cout<<"SMKN 2 Bandung"<<endl

### Output:

```
SMKN 2 Bandung  
SMKN 2 Bandung  
SMKN 2 Bandung  
SMKN 2 Bandung  
SMKN 2 Bandung  
-----  
Process exited after 0.04974 seconds with return value 0  
Press any key to continue . . .
```

### Penjelasan:

- Nilai awal perulangan adalah 1 (i=1)
- Do -> Awal pengulangan
  - Cetak pernyataan 1
  - Nilai i ditambah sehingga i=2
- Lakukan proses pengecekan, apakah nilai i (i=2) masih memenuhi i<=5
  - ya -> kerjakan bagian yang diulang (pernyataan 1) -> mencetak tulisan SMKN 2 Bandung
  - nilai i ditambah 1 -> i=3
- Lakukan proses pengecekan, apakah nilai i (i=3) masih memenuhi i<=5
  - ya -> kerjakan bagian yang diulang (pernyataan 1) -> mencetak tulisan SMKN 2 Bandung
  - nilai i ditambah 1 -> i=4
- Lakukan proses pengecekan, apakah nilai i (i=4) masih memenuhi i<=5

- ya -> kerjakan bagian yang diulang (pernyataan 1) -> mencetak tulisan SMKN 2 Bandung
- nilai i ditambah 1 -> i=5
- Lakukan proses pengecekan, apakah nilai i (i=5) masih memenuhi i<=5
  - ya -> kerjakan bagian yang diulang (pernyataan 1) -> mencetak tulisan SMKN 2 Bandung
  - nilai i ditambah 1 -> i=6
- lakukan proses pengecekan, apakah nilai i (i=6) masih memenuhi i<=5
  - tidak -> program keluar dari pengulangan dan kerjakan pernyataan 2 dan baris seterusnya.

### Perbedaan Do While dengan While

Perbedaan dengan perintah WHILE yaitu pada DO WHILE statement perulangannya dilakukan terlebih dahulu baru kemudian di cek kondisinya. Sedangkan WHILE kondisi dicek dulu baru kemudian statement perulangannya dijalankan. Akibat dari hal ini adalah dalam DO WHILE minimal terdapat 1x perulangan. Sedangkan WHILE dimungkinkan perulangan tidak pernah terjadi yaitu ketika kondisinya langsung bernilai FALSE.

### Contoh:

#### Program Do While

```
int main(){
int i;
i = 1;
do {
    cout<<"SMKN 2 Bandung"<<endl;
    i++;
}
while (i == 0);
}
```

#### Output

```
SMKN 2 Bandung
-----
Process exited after 0.04486 seconds with return value 0
Press any key to continue . . .
```

#### Program While

```
int main(){
int i;
i = 1;
while (i == 0){
    cout<<"SMKN 2 Bandung"<<endl;
    i++;
}
}
```

#### Output

```
-----
Process exited after 0.04983 seconds with return value 0
Press any key to continue . . .
```

## **6. Pemrograman Dasar menggunakan Tahapan *Computational Thinking***

*Computational thinking* atau berpikir komputasi merupakan sebuah pendekatan yang dapat membantu seseorang dalam menyelesaikan masalah secara sistematis dengan memfokuskan pada bagaimana suatu masalah dapat diselesaikan secara *step-by-step* dengan menggunakan serangkaian langkah-langkah yang terstruktur. Ada 4 tahapan dalam berpikir komputasi yaitu dekomposisi, abstraksi, pengenalan pola, dan desain algoritma.

Tahapan berpikir komputasi bisa menjadi salah satu pendekatan yang dapat membantu dalam menyelesaikan permasalahan pemrograman dasar. Dengan menggunakan tahapan berpikir komputasi, siswa dapat menyelesaikan masalah secara lebih sistematis. Adapun tahapan berpikir komputasi dalam pemrograman diantaranya:

- a) Define the Problem: Langkah pertama, kamu perlu mengidentifikasi masalah yang ingin diselesaikan.
- b) Break down the Problem: Pecah permasalahan menjadi beberapa bagian kecil agar lebih mudah dikelola serta memahami masalah.
- c) Use Abstraction: Hiraukan hal-hal tidak penting dalam sebuah permasalahan dan fokus pada informasi yang bisa membantu dalam menyelesaikan permasalahan.
- d) Identify Patterns: Setelah lebih fokus pada masalah, carilah suatu pola yang dapat membantu kamu untuk merancang solusi untuk menyelesaikan permasalahan.
- e) Design an Algorithm: Setelah memiliki rancangan solusi, selanjutnya adalah menggunakan pemahaman tentang masalah yang sudah kamu dapatkan untuk merancang sebuah algoritma untuk menyelesaikan permasalahannya. Dalam pemrograman, algoritma dapat berupa flowchart atau pseudocode.
- f) Implement the solution: Buat sebuah kode program untuk menyelesaikan masalah sesuai dengan algoritma yang sudah dirancang. Lalu uji program tersebut.

- g) Evaluate: tahap terakhir adalah evaluasi. Setelah solusi diuji, selanjutnya adalah melakukan evaluasi terhadap solusi yang telah dibuat. Kamu perlu mengevaluasi apakah solusi yang telah kamu buat sesuai dengan yang diharapkan, apakah solusi tersebut efisien, dan apakah ada cara lain yang lebih baik untuk menyelesaikan masalah tersebut.

### **Contoh:**

Buat program yang meminta pengguna memasukkan sebuah angka, kemudian menampilkan jumlah bilangan dari 1 hingga angka tersebut. Misalnya, jika pengguna memasukkan angka 5, maka program akan menampilkan  $1 + 2 + 3 + 4 + 5 = 15$ .

### **Penyelesaian:**

#### 1) Define the Problem

Berdasarkan soal, kita diminta untuk membuat program untuk menampilkan bilangan dari 1 hingga ke-n. Angka n didapat dari input user. Lalu pada bagian akhir, program harus menampilkan jumlah dari bilangan yang ditampilkan

#### 2) Break down the Problem


- a. Program menampilkan bilangan 1 hingga n
- b. N diperoleh dari inputan user
- c. Menjumlahkan bilangan yang ditampilkan

#### 3) Use Abstraction

- a. Batas awal adalah 1
- b. Melakukan perulangan hingga ke-n
- c. Mempunyai variabel tampungan untuk menghitung jumlah
- d. Setiap program melakukan perulangan, angka yang muncul akan masuk ke variabel tampung untuk dijumlahkan
- e. Kita tidak perlu tahu user akan menginputkan angka berapa

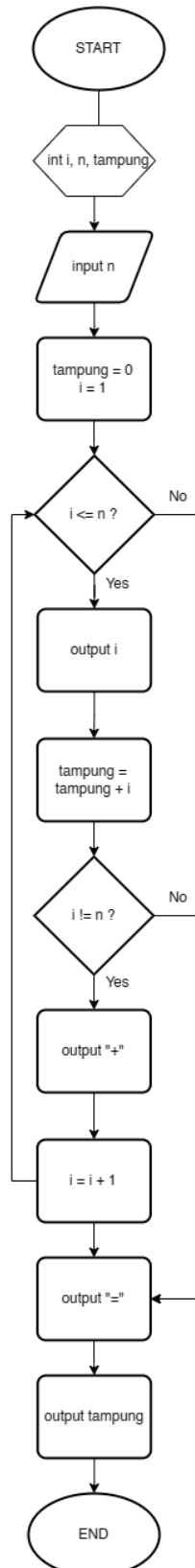
#### 4) Identify Patterns

Pola yang didapatkan antara lain:

- a. Program melakukan perulangan hingga batas akhir yaitu  $n$  ( $i \leq n$ ).
  - b. Setiap program melakukan perulangan, counter bertambah 1 ( $i = i + 1$  atau  $i++$ ).
  - c. Setiap program melakukan perulangan, bilangan yang muncul akan masuk ke variabel tampungan untuk dijumlahkan.
- 

## 5) Design an Algorithm

Selanjutnya membuat algoritma untuk mencari solusi dari permasalahan. Algoritma untuk persoalan di atas adalah:



## 6) Implement the solution

Selanjutnya, ubah algoritma kedalam sebuah kode program. Untuk persoalan di atas, apabila kita ubah kedalam bentuk program C++ akan menjadi:

```
int main (){  
    int i, n, tampung;  
    //input user  
    cout << "Masukkan angka: ";  
    cin >> n;  
    //deklarasi variabel tampung  
    tampung = 0;  
    for (i = 1; i <= n; i++){  
        //menampilkan angka i  
        cout << i;  
        //memasukkan angka i yang sudah di tampilkan ke  
        dalam variabel tampungan untuk dijumlahkan  
        tampung = tampung + i;  
        //menampilkan tanda "+"  
        if (i != n){  
            cout << " + ";  
        }  
    }  
    //menampilkan tanda "=" dan nilai yang sudah dijumlahkan  
    cout << " = " << tampung;  
}
```

## 7) Evaluate

Terakhir, uji coba kode program yang sudah dibuat dan lihat apakah hasilnya sudah sesuai atau belum. Harapan output apabila user memasukkan angka 5 adalah:  $1 + 2 + 3 + 4 + 5 = 15$ . Hasil output dari program di atas apabila user memasukkan angka 5 adalah:

```
Masukkan angka: 5
1 + 2 + 3 + 4 + 5 = 15
-----
Process exited after 1.551 seconds with return value 0
Press any key to continue . . .
```

## Kesimpulan:

Kode program yang dibuat sesuai dengan output yang diharapkan!



## E. Latihan

### 1. Perulangan For

Perhatikan soal dibawah ini!

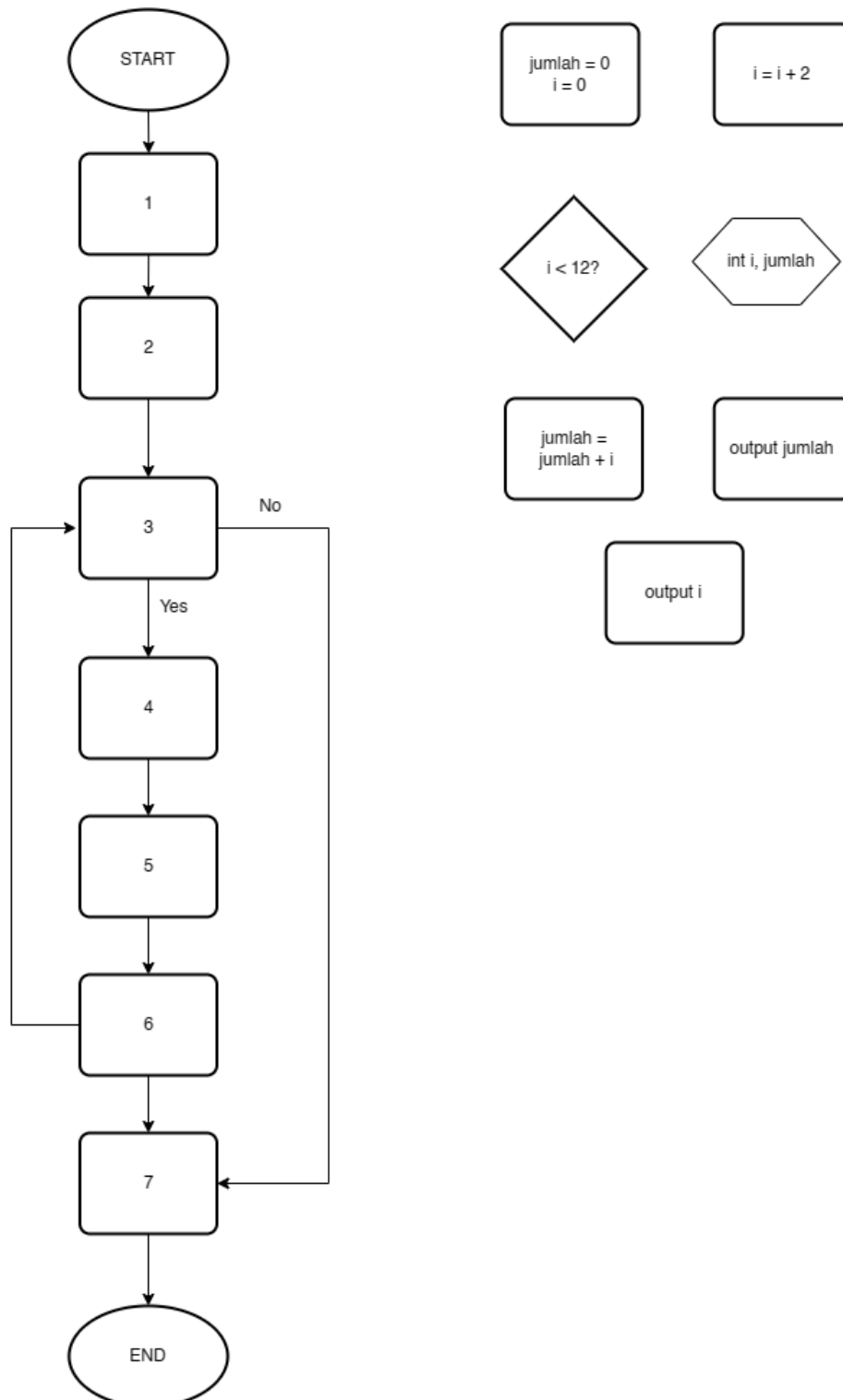
Buatlah program untuk menampilkan bilangan genap kurang dari 12 lalu hitung berapa jumlah dari bilangan genap yang ditampilkan!

- 1) Berdasarkan soal di atas, pilihlah informasi penting yang dapat menuntun kepada penyelesaian masalah (Pilihan jawaban bisa lebih dari 1)
  - a. Program harus menampilkan bilangan genap
  - b. Nilai awal = 0 dan nilai penaik (counter) = 2
  - c. Nilai awal = 1 dan nilai penaik (counter) = 2
  - d. Batas akhir = 12
  - e. Bilangan genap yang ditampilkan lalu dijumlahkan
- 2) Informasi apakah yang bisa kita abaikan dari soal di atas? (Pilihan jawaban bisa lebih dari 1)
  - a. Variabel yang digunakan harus variabel i dan j
  - b. Harus menampilkan bilangan genap
  - c. Harus menyediakan variabel tampungan untuk menghitung jumlah
  - d. Batas akhir pengulangan adalah 12
  - e. Menggunakan perintah For
- 3) Apakah pola penyelesaian di atas mirip dengan penyelesaian bagi masalah berikut?
  - a. Membuat program untuk menampilkan bilangan genap antara 100 sampai 125 tetapi dengan menampilkan dari nilai terbesar ke terkecil
    - a. TRUE            b) FALSE
  - b. Membuat program untuk menampilkan bilangan modulus yang habis atau tidak tersisa antara 50 sampai 70 lalu menjumlahkan bilangan yang muncul
    - a) TRUE            b) FALSE
  - c. Membuat program untuk menampilkan bilangan genap antara 100 sampai 125

a) TRUE

b) FALSE

4) Selesaikan flowchart dibawah ini dengan tahapan yang tepat!



## 2. Perulangan For Bersarang

Buatlah program untuk menampilkan gambar segitiga dalam bentuk bintang "\*" sesuai dengan input dari user. Misal, user menginput angka 3 maka outputnya

```
*  
* *  
* * *
```

- 1) Berdasarkan soal di atas, pilihlah informasi penting yang dapat menuntun kepada penyelesaian masalah (Pilihan jawaban bisa lebih dari 1)
  - a. Membuat program untuk menampilkan segitiga siku-siku
  - b. Terdapat input pada program
  - c. Output bintang "\*" ditampilkan sesuai dengan input dari user
  - d. User menginput angka 3
  - e. Perlu adanya pengulangan bersarang pada Program (kalang)
- 2) Informasi apakah yang bisa kita abaikan dari soal di atas? (Pilihan jawaban bisa lebih dari 1)
  - a. Angka inputan user
  - b. Jenis perulangan yang digunakan
  - c. Program dibuat untuk menampilkan segitiga sesuai input dari user
  - d. Terdapat 2 kali perulangan pada program
  - e. Segitiga yang dibuat ditampilkan menggunakan simbol bintang "\*".
- 3) Apakah pola penyelesaian di atas mirip dengan penyelesaian bagi masalah berikut?
  - a. Ahmad diminta untuk membuat program menampilkan segitiga siku-siku dengan panjang berukuran 3 cm dan tinggi berukuran 5 cm.
    - a) TRUE                      b) FALSE
  - b. Bimo diminta untuk membuat program menampilkan hasil luas dan keliling segitiga siku-siku. Nilai panjang dan tinggi didapat dari input user.
    - a) TRUE                      b) FALSE

c. Siswa diminta untuk membuat program menampilkan bilangan berpangkat hingga ke-N. N merupakan bilangan yang diambil dari input user sehingga output program akan seperti pada tampilan dibawah ini.

(Misal user input angka 5).

Output:

1

2 2

3 3 3

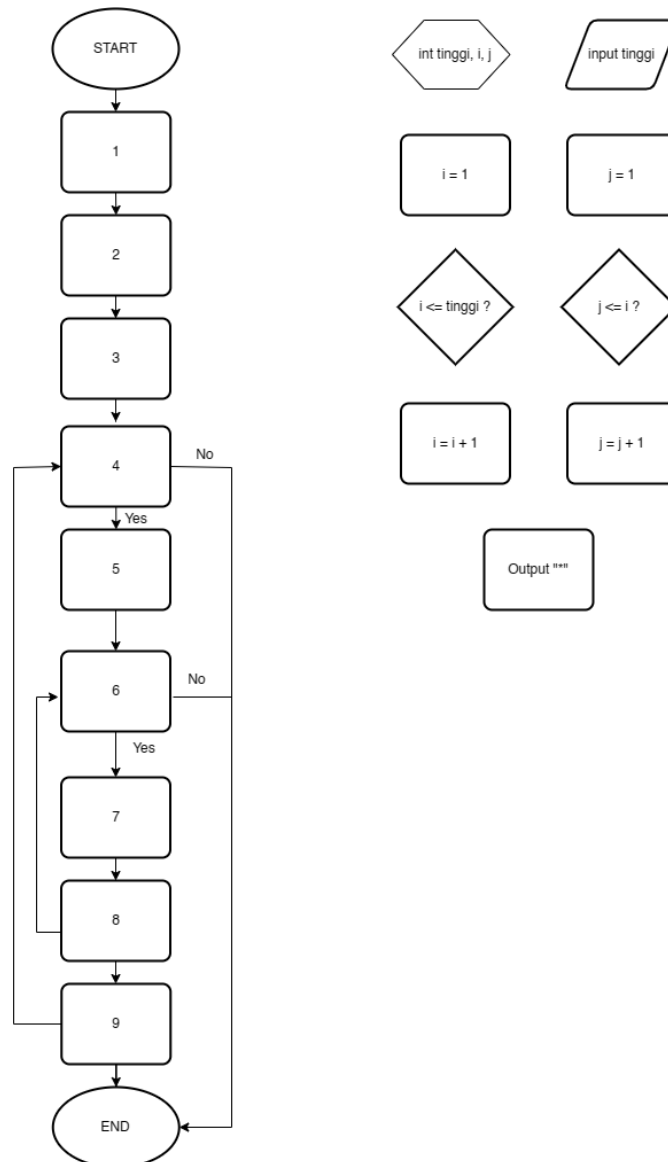
4 4 4 4

5 5 5 5 5

a) TRUE

b) FALSE

4) Selesaikan flowchart dibawah ini dengan tahapan yang tepat!



### 3. Perulangan While

Buatlah program pengulangan while untuk menampilkan bilangan N faktorial. Nilai N didapat dari inputan user. Program harus bisa menampilkan N faktorial dan menghitung jumlah dari faktorial tersebut.

Contoh:  $2! = 2 \times 1 = 2$ .

- 1) Berdasarkan soal di atas, pilihlah informasi penting yang dapat menuntun kepada penyelesaian masalah (Pilihan jawaban bisa lebih dari 1)
  - a. Program harus memiliki input untuk menampilkan dan menjumlahkan bilangan faktorial
  - b. Rumus bilangan faktorial adalah  $N! = N \times (N-1)$
  - c. Terdapat variabel tampungan untuk menghitung jumlah
  - d. Program pengulangan menggunakan struktur kode While
  - e. Inputan user adalah angka 2
- 2) Informasi apakah yang bisa kita abaikan dari soal di atas? (Pilihan jawaban bisa lebih dari 1)
  - a. User menginputkan angka 2
  - b. Jenis variabel yang digunakan pada program
  - c. Program harus bisa menampilkan bilangan faktorial
  - d. Program menampilkan dan menghitung bilangan faktorial sesuai dengan inputan user
  - e. Program harus bisa menjumlahkan bilangan faktorial
- 3) Apakah pola penyelesaian di atas mirip dengan penyelesaian bagi masalah berikut?
  - a. Amri memiliki sepatu bermerk Adadas, Niko, dan Pans. Untuk menyimpan ketiga sepatu tersebut, Amri memiliki 6 kemungkinan untuk menyimpan sepatu dengan urutan yang berbeda. Ke-6 kemungkinan susunan sepatu tersebut adalah sebagai berikut

Adadas	–	Niko	–	Adadas	–	Pans	–	Niko	–	Adadas	–
Pans				Niko				Pans			

Niko - Pans - Pans - Adadas - Pans - Niko -  
Adadas Niko Adadas

Apabila Amri membeli sepatu dengan 2 merk yang baru, buatlah program untuk menghitung banyaknya kemungkinan susunan sepatu yang dimiliki oleh Amri?

a) TRUE      b) FALSE

- b. Siswa diminta untuk membuat program menampilkan bilangan berpangkat hingga ke-N. N merupakan bilangan yang diambil dari input user sehingga output program akan seperti pada tampilan dibawah ini.

(Misal user input angka 5), Output:

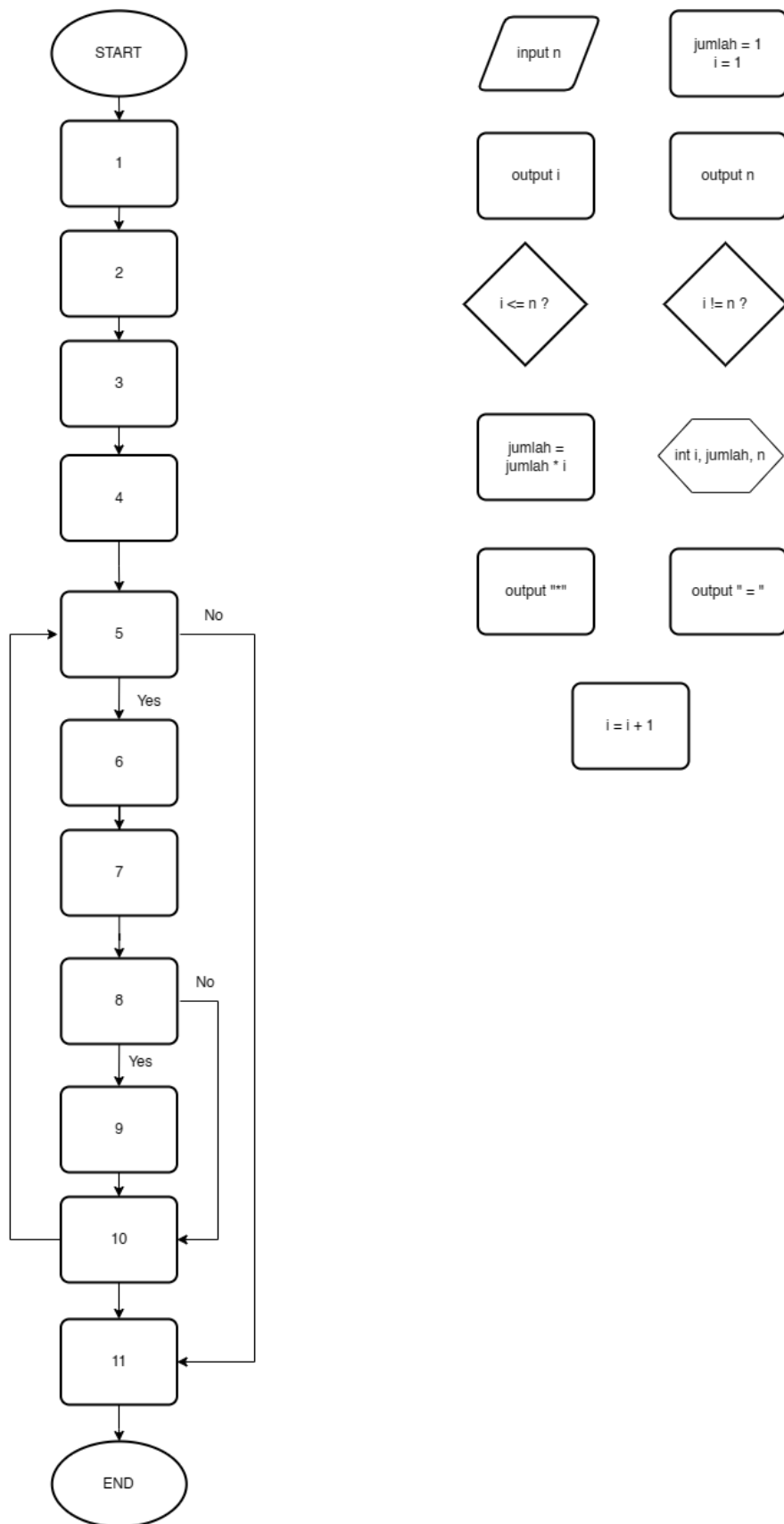
1  
2 2  
3 3 3  
4 4 4 4  
5 5 5 5 5

a) TRUE      b) FALSE

- c. Buatlah program untuk menampilkan dan menjumlahkan 10!.

a) TRUE      b) FALSE

4) Selesaikan flowchart dibawah ini dengan tahapan yang tepat!



#### 4. Perulangan Do – While

Mira mahasiswa komputer di salah satu universitas di Bandung diminta untuk membuat sebuah program menu sederhana untuk bisa menghitung luas dan keliling sebuah persegi. Dalam program tersebut, user akan diminta untuk menginputkan besar sisi dari persegi. Program akan terus berulang selama user tidak memilih opsi exit. Bantulah mira untuk membuat program tersebut!

- 1) Berdasarkan soal di atas, pilihlah informasi penting yang dapat menuntun kepada penyelesaian masalah (Pilihan jawaban bisa lebih dari 1)
  - a. Program akan terdiri dari 3 menu yaitu menghitung luas persegi, menghitung keliling persegi, dan exit.
  - b. Program menggunakan konsep perulangan Do – While
  - c. Rumus luas persegi = sisi \* sisi dan rumus keliling persegi = 4 \* sisi
  - d. Program memiliki user input untuk mengambil data dari besar sisi persegi
  - e. Program akan terus berulang selama user tidak memilih menu exit
- 2) Informasi apakah yang bisa kita abaikan dari soal di atas? (Pilihan jawaban bisa lebih dari 1)
  - a. Rumus luas dan keliling persegi
  - b. Mira merupakan mahasiswa komputer di salah satu Universitas di Bandung
  - c. Besar sisi yang diinput oleh user
  - d. Program harus memiliki input user untuk mengambil data sisi persegi
  - e. Program akan terus berulang selama user tidak memilih menu exit
- 3) Apakah pola penyelesaian di atas mirip dengan penyelesaian bagi masalah berikut?
  - a. Buatlah program menu menggunakan konsep perulangan do-while untuk menghitung luas dan keliling segitiga. Besar alas dan tinggi tergantung input user.
    - a) TRUE
    - b) FALSE



b. Buatlah program untuk menghitung luas dan keliling persegi dengan panjang sebesar 5 cm.

a) TRUE                      b) FALSE

c. Buatlah program untuk menghitung luas atau keliling persegi panjang. Dalam program, user bisa memilih ingin menghitung luas atau keliling dan menginputkan panjang juga lebar dari persegi panjang. Program akan berhenti setelah user memilih ingin menghitung luas atau keliling dari persegi panjang

a) TRUE                      b) FALSE

4) Selesaikan flowchart dibawah ini dengan tahapan yang tepat!

