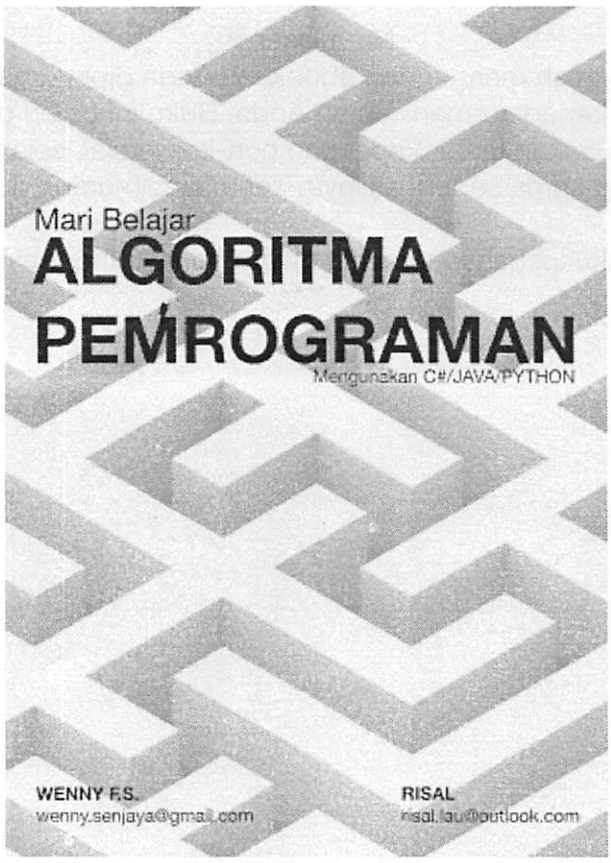
02


Mari Belajar
ALGORITMA
PEMROGRAMAN
Menggunakan C#/JAVA/PYTHON
WENNY F.S.
wenny.senjaya@gmail.com
RISAL
risal.lau@outlook.com

Mari Belajar Algoritma Pemrograman Menggunakan C#/JAVA/Python

Published by Risal at Smashwords

ISBN 9781370293292

This book is only availabe in Indonesian.

Buku ini hanya tersedia dalam bahasa Indonesia.

***

Kata Pengantar

Puji dan syukur kepada Tuhan Yang Maha Esa, sehingga penulis dapat menyelesaikan penyusunan buku ini dengan baik.

Buku ini disusun untuk menuntun para pemula yang ingin mempelajari pemrograman. Penulis menyajikan konsep-konsep dasar pemrograman yang dikemas dengan bahasa yang sederhana dan disertai dengan contoh-contoh penyelesaian kasus. Penyelesaian kasus disajikan dalam tiga bahasa pemrograman yaitu C#, JAVA, dan Python, dengan harapan agar pembaca tidak terpaku pada satu bahasa pemrogramaan saja dan lebih mudah ketika mempelajari bahasa pemrograman lain.

Akhir kata, Penulis ingin menyampaikan Terima kasih kepada berbagai pihak yang telah mendukung penulis dalam pembuatan dan penyelesaian buku ini.

Bandung, Juli 2017

Penulis

***

Algoritma dan Program

Ketika belajar pemrograman pertama kali, ada beberapa jargon (kata asing) yang digunakan. Beberapa jargon umum yang digunakan akan dijelaskan pada bab pertama, sedangkan sisanya akan dijelaskan pada bab-bab yang terkait. Dua jargon pertama yang sering muncul adalah algoritma dan program.

Apa itu algoritma? Algoritma adalah sekumpulan langkah yang disusun untuk menyelesaikan sebuah pekerjaan. Algoritma yang baik memiliki jumlah langkah pengerjaan yang minim.

Bagaimana dengan program? Program adalah sekumpulan perintah (instruksi) yang dimengerti oleh komputer. Orang yang membuat program dikenal dengan nama programmer (dibeberapa buku dikenal juga dengan nama developer). Sedangkan kegiatan untuk membuat program untuk menyelesaikan sebuah masalah dikenal dengan istilah pemrograman.

Jadi dapat disimpulkan bahwa algoritma bersifat universal, sedangkan program sudah lebih spesifik pada instruksi yang dikenali oleh komputer. Algoritma biasanya dituliskan dalam bentuk bahasa yang lebih dimengerti oleh manusia atau dapat juga dituliskan dalam notasi matematik (dikenal dengan metode formal).

Bagaimana dengan contoh dari algoritma itu sendiri? Berikut ini adalah ilustrasi/contoh solusi (algoritma) dari masalah memasak mie instan menggunakan bahasa natural (bahasa manusia).

1. Buka bungkus mie instan.
2. Didihkan air.
3. Masukan mie instan ke dalam air mendidih.
4. Masukan bumbu ke dalam wadah.
5. Masukan mie instan yang telah masak ke dalam wadah.
6. Aduk mie instan dengan bumbu.

Bagaimana bilamana diminta untuk menuliskan algoritma menukar isi dua buah gelas? Berikut adalah jawabannya. Asumsi dua buah gelas diberi nama gelas A dan gelas B. Gelas C adalah gelas bantuan untuk memindahkan isi gelas tersebut (gelas A dan gelas B).

1. Pindahkan isi gelas A ke gelas C.
2. Pindahkan isi gelas B ke gelas A.
3. Pindahkan isi gelas C ke gelas B.

***

Paradigma Pemrograman

Paradigma pemrograman adalah cara pandang dalam membuat sebuah program. Bahasa pemrograman dapat mendukung satu atau lebih paradigma pemrograman. Dua paradigma pemrograman yang populer dan sering digunakan adalah paradigma prosedural dan paradigma berorientasi objek.

Paradigma prosedural (imperatif) adalah paradigma pemrograman yang membagi masalah (besar) menjadi kumpulan masalah kecil (sub masalah). Tiap masalah kecil ini diselesaikan menggunakan sekumpulan langkah yang (biasanya) dibuat dalam sebuah prosedur.

Berbeda dengan paradigma prosedural, paradigma berorientasi objek membagi masalah menjadi kumpulan objek yang saling berinteraksi/berkolaborasi satu dengan yang lainnya. Objek ini adalah instans dari kelas. Kelas adalah sebuah pola yang berisi kumpulan atribut dan operasi. Atribut dan operasi ini yang nantinya diakses oleh objek/kelas lainnya untuk menyelesaikan sebuah masalah.

Selain dua paradigma ini, dikenal juga beberapa paradigma pemrograman lainnya yang cocok dalam menyelesaikan masalah dengan sudut pandang yang berbeda. Paradigma yang dimaksud adalah antara lain deklaratif, logika, simbolik, dan berorientasi aspek.

***

Hi, Dunia!

Seperti buku pemrograman pada umumnya, kode pertama kali dibuat adalah "Hello World". Buku ini akan menggunakan tiga buah bahasa pemrograman modern, yaitu C# (dibaca C Sharp), JAVA, dan Python. Dengan menggunakan tiga bahasa pemrograman modern ini diharapkan dapat memudahkan transisi antara ketiga bahasa pemrograman.

```
class Program
{
 static void Main(string[] args)
 {
 System.Console.WriteLine("Hi Dunia!");

// Keluaran Program
 // Hi Dunia!
 }
}
```

Bahasa pemrograman C# menggunakan method WriteLine dari kelas Console untuk menampilkan teks ke layar konsol. Kelas Console ada pada namespace System. Namespace kata kunci pada bahasa pemrograman C# yang bertujuan untuk mengelompokan kumpulan kelas. Nama kelas dalam sebuah namespace harus unik (tidak boleh sama).

```
public class Program {
 public static void main(String[] args) {
 System.out.println("Hi Dunia!");

// Keluaran Program
 // Hi Dunia!
 }
}
```

Bahasa pemrograman JAVA menggunakan method println yang dapat diakses melalui variabel out di kelas System. Berbeda sedikit dengan bahasa C#, pada bahasa JAVA System adalah sebuah kelas. JAVA juga mengenal konsep yang serupa dengan namespace di C#, pada JAVA dikenal dengan nama package.

```
def main():
 print("Hi Dunia!")

 # Keluaran Program
 # Hi Dunia!

if __name__ == '__main__':
 main()
```

Bahasa pemrograman PYTHON menggunakan method print untuk menampilkan keluaran program. Berbeda dengan bahasa pemrograman C# dan JAVA, pada bahasa pemrograman PYTHON setiap akhir statement tidak diakhiri dengan ; (titik koma).

***

Tipe Data

Tipe data (data type) atau biasa disingkat menjadi tipe (type) adalah representasi data dalam komputer. Tipe digunakan oleh bahasa pemrograman untuk memaksakan dan verifikasi data atau operasi. Tiap bahasa pemrograman memiliki beberapa tipe data dasar dan tipe data spesifik. Berikut adalah beberapa jenis tipe data yang ada pada bahasa pemrograman.

1. Bilangan logika (boolean).
2. Bilangan bulat.
3. Bilangan riil.
4. Karakter.
5. String.

***

Bilangan Logika

Tipe data bilangan logika (boolean) adalah tipe data yang digunakan untuk menyimpan hasil dari ekspresi logika (boolean). Tipe data boolean memiliki dua kemungkinan nilai yaitu true atau false. Kata kunci bool digunakan oleh bahasa pemrograman C# dan Python untuk menggunakan tipe data ini, sedangkan bahasa pemrograman JAVA menggunakan kata kunci boolean.

Tipe data boolean mengenal tiga buah operator, yaitu operator NOT, operator AND, dan operator OR. Tabel berikut adalah ilustrasi hasil dari ketiga operator tersebut.

x	NOT x
false	true
true	false

Operator NOT atau negasi mengembalikan nilai true bilamana nilai operan bernilai false dan sebaliknya.

x	y	x AND y
false	false	false
false	true	false
true	false	false
true	true	true

Operator AND mengembalikan nilai true jika kedua nilai operan bernilai true.

x	y	x OR y
false	false	false
false	true	true
true	false	true
true	true	true

Operator OR mengembalikan nilai false jika kedua nilai operan bernilai false. Kode program berikut digunakan untuk mendemonstrasikan ketiga operator yang telah dibahas. Berikut ini adalah contoh kode program C# yang mendemonstrasikan bilangan logika dan operatornya.

```
using System;

class Program
{
 static void Main(string[] args)
 {
 bool b0 = true;
 bool b1 = 3 < 7;
 bool b2 = 6.2 <= 6.5;
 bool b3 = "budi" != "wati";

 Console.WriteLine("b0 = " + b0);
 Console.WriteLine("b1 = " + b1);
 Console.WriteLine("b2 = " + b2);
```

```
 Console.WriteLine("b3 = " + b3);

 // Operator AND
 Console.WriteLine("b0 AND b1 = " + (b0 && b1));

 // Operator OR
 Console.WriteLine("b1 OR b2 = " + (b1 || b2));

 // Operator NOT
 Console.WriteLine("NOT b3 = " + (!b3));

 // Keluaran Program
 // b0 = True
 // b1 = True
 // b2 = True
 // b3 = True
 // b0 AND b1 = True
 // b1 OR b2 = True
 // NOT b3 = False
 }
}
```

Berikut ini adalah kode program serupa dengan menggunakan bahasa pemrograman JAVA.

```
public class Program {
 public static void main(String[] args) {
 boolean b0 = true;
 boolean b1 = 3 < 7;
 boolean b2 = 6.2 <= 6.5;
 boolean b3 = "budi" != "wati";

 System.out.println("b0 = " + b0);
 System.out.println("b1 = " + b1);
 System.out.println("b2 = " + b2);
 System.out.println("b3 = " + b3);

 // Operator AND
 System.out.println("b0 AND b1 = " + (b0 && b1));

 // Operator OR
 System.out.println("b1 OR b2 = " + (b1 || b2));

 // Operator NOT
 System.out.println("NOT b3 = " + (!b3));

 // Keluaran Program
 // b0 = True
 // b1 = True
 // b2 = True
 // b3 = True
 // b0 AND b1 = True
 // b1 OR b2 = True
 // NOT b3 = False
 }
}
```

Sedangkan berikut ini adalah kode program dalam bahasa pemrograman Python.

```
def main():
 b0 = True
 b1 = 3 < 7
```

```
    b2 = 6.2 <= 6.5
    b3 = "budi" != "wati"

    print("b0 =", b0)
    print("b1 =", b1)
    print("b2 =", b2)
    print("b3 =", b3)

    # Operator AND
    print("b0 AND b1 =", (b0 and b1))

    # Operator OR
    print("b1 OR b2 =", (b1 or b2))

    # Operator NOT
    print("NOT b3 =", not b3)

    # Keluaran Program
    # b0 = True
    # b1 = True
    # b2 = True
    # b3 = True
    # b0 AND b1 = True
    # b1 OR b2 = True
    # NOT b3 = False

if __name__ == '__main__':
    main()
```

***

Bilangan Bulat

Tipe data bilangan bulat (integer) adalah tipe data yang digunakan untuk memodelkan bilangan tidak berkoma, contoh: 0, 12, -3, dan 79. Tergantung bahasa pemrograman yang digunakan operasi yang dapat dilakukan pada tipe integer dan variasi tipe data yang termasuk dalam kategori integer berbeda satu dengan yang lainnya.

Bahasa pemrograman C# membagi tipe data integer dibagi menjadi dua, empat buah tipe data integer bertanda (sbyte, short, int, dan long) dan empat buah tipe data integer tidak bertanda (byte, ushort, uint, dan ulong). Integer bertanda adalah integer yang memiliki nilai negatif dan positif, sedangkan integer tidak bertanda hanya memiliki nilai positif saja.

Operator	Arti	Contoh	Hasil
+	Tambah	12+5	17
-	Kurang	12-5	7
*	Kali	12*5	60
/	Bagi	12/5	2
%	Sisa bagi	12%5	2
++	Tambah satu	num=5; num++;	6
--	Kurang satu	num=5; num--;	4
<<	Shift bit kiri	1<<2	4
>>	Shift bit kanan	3>>1	1

Berikut adalah contoh kode program C# yang bertujuan untuk mendemonstrasikan operator aritmatika pada tabel sebelumnya.

```
using System;

class Program
{
 static void Main(string[] args)
 {
 int n1 = 5, n2 = 12;
 int h1 = n1 + n2;
 int h2 = n2 - n1;
 int h3 = n1 * n2;
 int h4 = n2 / n1;
 int h5 = n2 % n1;
 int h6 = n1 << 2;
 int h7 = n2 >> 2;

 Console.WriteLine("{0} + {1} = {2}", n1, n2, h1);
 Console.WriteLine("{0} - {1} = {2}", n2, n1, h2);
 Console.WriteLine("{0} * {1} = {2}", n1, n2, h3);
 Console.WriteLine("{0} / {1} = {2}", n2, n1, h4);
 Console.WriteLine("{0} % {1} = {2}", n2, n1, h5);
 Console.WriteLine("{0} << 2 = {1}", n1, h6);
 Console.WriteLine("{0} >> 2 = {1}", n2, h7);

 // Post-increment
 Console.WriteLine("{0}++ = {1}", n1, n1++);

 // Post-decrement
```

```
 Console.WriteLine("{0}-- = {1}", n2, n2--);

 // Keluaran Program
 // 5 + 12 = 17
 // 12 - 5 = 7
 // 5 * 12 = 60
 // 12 / 5 = 2
 // 12 % 5 = 2
 // 5 << 2 = 20
 // 12 >> 2 = 3
 // 5++ = 6
 // 12-- = 11
 }
}
```

Bahasa pemrograman JAVA menggunakan pendekatan sedikit berbeda dengan bahasa pemrograman C#. Bahasa pemrograman JAVA hanya memiliki empat tipe data integer yaitu byte, short, int, dan long. Operator aritmatika pada bahasa pemrograman JAVA sama dengan bahasa pemrograman C#. Kode program berikut mendemonstrasikan penggunaan operator aritmatika menggunakan bahasa JAVA.

```
public class Program {
 public static void main(String[] args) {
 int n1 = 5, n2 = 12;
 int h1 = n1 + n2;
 int h2 = n2 - n1;
 int h3 = n1 * n2;
 int h4 = n2 / n1;
 int h5 = n2 % n1;
 int h6 = n1 << 2;
 int h7 = n2 >> 2;

 System.out.println(n1 + " + " + n2 + " = " + h1);
 System.out.println(n2 + " - " + n1 + " = " + h2);
 System.out.println(n1 + " * " + n2 + " = " + h3);
 System.out.println(n2 + " / " + n1 + " = " + h4);
 System.out.println(n2 + " % " + n1 + " = " + h5);
 System.out.println(n1 + " << 2 = " + h6);
 System.out.println(n2 + " >> 2 = " + h7);

 // Post-increment
 System.out.println(n1 + "++ = " + (n1++));

 // Post-decrement
 System.out.println(n2 + "-- = " + (n2--));

 // Keluaran Program
 // 5 + 12 = 17
 // 12 - 5 = 7
 // 5 * 12 = 60
 // 12 / 5 = 2
 // 12 % 5 = 2
 // 5 << 2 = 20
 // 12 >> 2 = 3
 // 5++ = 6
 // 12-- = 11
 }
}
```

Operator aritmatika pada bahasa pemrograman Python pada dasarnya sama dengan bahasa pemrograman C# dan JAVA. Pada Python terdapat beberapa tambahan operator

aritmatika, berikut beberapa operator aritmatika tambahan pada Python.

Operator	Arti	Contoh	Hasil
**	Pangkat	2**3	8
//	Bagi(hasil bilangan bulat)	13//2	6

Berikut adalah kode program serupa dalam bahasa pemrograman Python.

```
def main():
 n1 = 5
 n2 = 12
 h1 = n1 + n2
 h2 = n2 - n1
 h3 = n1 * n2
 h4 = n2 // n1
 h5 = n2 % n1
 h6 = n1 << 2
 h7 = n2 >> 2

 print(n1 , "+" , n2 , "=" , h1)
 print(n2 , "-" , n1 , "=" , h2)
 print(n1 , "*" , n2 , "=" , h3)
 print(n2 , "//" , n1 , "=" , h4)
 print(n2 , "%" , n1 , "=" , h5)
 print(n1 , "<< 2" , "=" , h6)
 print(n2 , ">> 2" , "=" , h7)

 # Post-increment
 print(n1 , "+=1 =", end="")
 n1+=1
 print(n1)

 # Pre-decrement
 print(n2 , "-=1 =" , end="")
 n2-=1
 print(n2)

 # Keluaran Program
 # 5 + 12 = 17
 # 12 - 5 = 7
 # 5 + 12 = 60
 # 12 // 5 = 2
 # 12 % 5 = 2
 # 5 << 2 = 20
 # 12 >> 2 = 3
 # 5+=1 = 6
 # 12-=1 = 11

if __name__ == '__main__':
 main()
```

Berbeda dengan operator artitmatika, operator relasional digunakan untuk membandingkan dua buah operan (nilai). Hasil dari operator relasional bertipe boolean (true atau false). Operator relasional pada bahasa pemrograman C#, bahasa pemrograman JAVA, dan bahasa pemrograman Phyton sama.

Operator	Arti	Contoh	Hasil
<	Lebih kecil	12 < 5	false

<=	Lebih kecil atau sama dengan	12 <= 5	false
>	Lebih besar	12 > 5	true
>=	Lebih besar atau sama dengan	12 >= 5	true
==	Sama dengan	12 == 5	false
!=	Tidak sama dengan	12 != 5	true

Kode program C# berikut mendemonstarikan penggunaan operator relasional.

```
using System;

class Program
{
 static void Main(string[] args)
 {
 int n1 = 5, n2 = 12;

 Console.WriteLine("{0} < {1}? {2}", n1, n2, (n1 < n2));
 Console.WriteLine("{0} <= {1}? {2}", n1, n2, (n1 <= n2));
 Console.WriteLine("{0} > {1}? {2}", n1, n2, (n1 > n2));
 Console.WriteLine("{0} >= {1}? {2}", n1, n2, (n1 >= n2));
 Console.WriteLine("{0} == {1}? {2}", n1, n2, (n1 == n2));
 Console.WriteLine("{0} != {1}? {2}", n1, n2, (n1 != n2));

 // Keluaran Program
 // 5 < 12? True
 // 5 <= 12? True
 // 5 > 12? False
 // 5 >= 12? False
 // 5 == 12? False
 // 5 != 12? True
 }
}
```

Berikut adalah kode program serupa dalam bahasa pemrograman JAVA.

```
public class Program {
 public static void main(String[] args) {
 int n1 = 5, n2 = 12;

 System.out.println(n1 + " < " + n2 + "? " + (n1 < n2));
 System.out.println(n1 + " <= " + n2 + "? " + (n1 <= n2));
 System.out.println(n1 + " > " + n2 + "? " + (n1 > n2));
 System.out.println(n1 + " >= " + n2 + "? " + (n1 >= n2));
 System.out.println(n1 + " == " + n2 + "? " + (n1 == n2));
 System.out.println(n1 + " != " + n2 + "? " + (n1 != n2));

 // Keluaran Program
 // 5 < 12? True
 // 5 <= 12? True
 // 5 > 12? False
 // 5 >= 12? False
 // 5 == 12? False
 // 5 != 12? True
 }
}
```

Sedangkan kode program berikut adalah kode program dalam bahasa Python.

```
def main():
```

```
    num1 = 5
    num2 = 12

    print(num1 , "<" , num2 , "?" , (num1 < num2))
    print(num1 , "<=" , num2 , "?" , (num1 <= num2))
    print(num1 , ">" , num2 , "?" , (num1 > num2))
    print(num1 , ">=" , num2 , "?" , (num1 >= num2))
    print(num1 , "==" , num2 , "?" , (num1 == num2))
    print(num1 , "!=" , num2 , "?" , (num1 != num2))

    # Keluaran Program
    # 5 < 12? True
    # 5 <= 12? True
    # 5 > 12? False
    # 5 >= 12? False
    # 5 == 12? False
    # 5 != 12? True

if __name__ == '__main__':
    main()
```

***

Bilangan Riil

Tipe data bilangan riil (real) adalah tipe data yang digunakan untuk memodelkan bilangan berkoma (desimal), contoh: 3.1415, -2.3, 4.0, dan -0.333333. Tipe data ini dikenal juga dengan nama floating point. Operasi pada tipe data riil biasanya lebih terbatas daripada operasi pada tipe data integer.

Bahasa pemrograman C# dan JAVA memiliki dua tipe data yang termasuk kategori ini. Kedua tipe data tersebut adalah float dan double. Perbedaan antara float dan double hanya pada tingkat keakuratan (presisi). Tipe data double memiliki presisi lebih tinggi dari pada tipe data float.

Operator	Arti	Contoh	Hasil
+	Tambah	11.7 + 0.3	12
-	Kurang	11.7 - 0.3	11.4
*	Kali	11.7 * 0.3	3.51
/	Bagi	11.7 / 0.3	39.0
%	Sisa bagi	11.7 % 11	0.7

Berikut ini adalah contoh kode program dalam bahasa pemrograman C#.

```
using System;

class Program
{
 static void Main(string[] args)
 {
 double n1 = 3.14, n2 = 9.8;
 double h1 = n1 + n2;
 double h2 = n2 - n1;
 double h3 = n1 * n2;
 double h4 = n2 / n1;
 double h5 = n2 % n1;

 Console.WriteLine("{0} + {1} = {2}", n1, n2, h1);
 Console.WriteLine("{0} - {1} = {2}", n2, n1, h2);
 Console.WriteLine("{0} * {1} = {2}", n1, n2, h3);
 Console.WriteLine("{0} / {1} = {2}", n2, n1, h4);
 Console.WriteLine("{0} % {1} = {2}", n2, n1, h5);

 // Keluaran Program
 // 3.14 + 9.8 = 12.94
 // 9.8 - 3.14 = 6.66
 // 3.14 * 9.8 = 30.772
 // 9.8 / 3.14 = 3.12101910828026
 // 9.8 % 3.14 = 0.38
 }
}
```

Berikut ini adalah contoh kode program dalam bahasa pemrograman JAVA.

```
public class Program {
 public static void main(String[] args) {
 double n1 = 3.14, n2 = 9.8;
 double h1 = n1 + n2;
 double h2 = n2 - n1;
```

```
 double h3 = n1 * n2;
 double h4 = n2 / n1;
 double h5 = n2 % n1;

 System.out.println(n1 + " + " + n2 + " = " + h1);
 System.out.println(n2 + " - " + n1 + " = " + h2);
 System.out.println(n1 + " * " + n2 + " = " + h3);
 System.out.println(n2 + " / " + n1 + " = " + h4);
 System.out.println(n2 + " % " + n1 + " = " + h5);

 // Keluaran Program
 // 3.14 + 9.8 = 12.940000000000001
 // 9.8 - 3.14 = 6.66
 // 3.14 * 9.8 = 30.772000000000002
 // 9.8 / 3.14 = 3.121019108280255
 // 9.8 % 3.14 = 0.38000000000000034
 }
}
```

Berikut ini adalah contoh kode program dalam bahasa pemrograman Python.

```
def main():
 n1 = 3.14
 n2 = 9.8
 h1 = n1 + n2
 h2 = n2 - n1
 h3 = n1 * n2
 h4 = n2 / n1
 h5 = n2 % n1

 print(n1 , "+" , n2 , "=" , h1)
 print(n2 , "-" , n1 , "=" , h2)
 print(n1 , "*" , n2 , "=" , h3)
 print(n2 , "/" , n1 , "=" , h4)
 print(n2 , "%" , n1 , "=" , h5)

 # Keluaran Program
 # 3.14 + 9.8 = 12.940000000000001
 # 9.8 - 3.14 = 6.66
 # 3.14 * 9.8 = 30.772000000000002
 # 9.8 / 3.14 = 3.121019108280255
 # 9.8 % 3.14 = 0.38000000000000034

if __name__ == '__main__':
 main()
```

Sama seperti bilangan bulat, bilangan riil juga memiliki beberapa operator relasional yang dapat digunakan untuk membandingkan dua buah nilai. Berikut ini adalah tabel dari operator relasional untuk ketiga bahasa pemrograman (C#, JAVA, dan Python).

Operator	Arti	Contoh	Hasil
<	Lebih kecil	11.7 < 0.3	false
<=	Lebih kecil atau sama dengan	11.7 <= 0.3	false
>	Lebih besar	11.7 > 0.3	true
>=	Lebih besar atau sama dengan	11.7 >= 0.3	true
!=	Tidak sama dengan	11.7 != 0.3	true

Berikut ini adalah contoh program dalam bahasa C# yang menggunakan operator

relasional.

```
using System;

class Program
{
 static void Main(string[] args)
 {
 double n1 = 3.14, n2 = 9.8;

 Console.WriteLine("{0} < {1}? {2}", n1, n2, (n1 < n2));
 Console.WriteLine("{0} <= {1}? {2}", n1, n2, (n1 <= n2));
 Console.WriteLine("{0} > {1}? {2}", n1, n2, (n1 > n2));
 Console.WriteLine("{0} >= {1}? {2}", n1, n2, (n1 >= n2));
 Console.WriteLine("{0} != {1}? {2}", n1, n2, (n1 != n2));

 // Keluaran Program
 // 3.14 < 9.8? True
 // 3.14 <= 9.8? True
 // 3.14 > 9.8? False
 // 3.14 >= 9.8? False
 // 3.14 != 9.8? True
 }
}
```

Berikut ini adalah contoh program dalam bahasa JAVA.

```
public class Program {
 public static void main(String[] args) {
 int n1 = 3.14, n2 = 9.8;

 System.out.println(n1 + " < " + n2 + "? " + (n1 < n2));
 System.out.println(n1 + " <= " + n2 + "? " + (n1 <= n2));
 System.out.println(n1 + " > " + n2 + "? " + (n1 > n2));
 System.out.println(n1 + " >= " + n2 + "? " + (n1 >= n2));
 System.out.println(n1 + " != " + n2 + "? " + (n1 != n2));

 // Keluaran Program
 // 3.14 < 9.8? True
 // 3.14 <= 9.8? True
 // 3.14 > 9.8? False
 // 3.14 >= 9.8? False
 // 3.14 != 9.8? True }
}
```

Berikut ini adalah contoh program dalam bahasa Python.

```
def main():
 n1=3.14
 n2=9.8

 print(n1 , "<" , n2 , "?" , (n1 < n2))
 print(n1 , "<=" , n2 , "?" , (n1 <= n2))
 print(n1 , ">" , n2 , "?" , (n1 > n2))
 print(n1 , ">=" , n2 , "?" , (n1 >= n2))
 print(n1 , "!=" , n2 , "?" , (n1 != n2))

 # Keluaran Program
 # 3.14 < 9.8? True
 # 3.14 <= 9.8? True
 # 3.14 > 9.8? False
 # 3.14 >= 9.8? False
```

```
 # 3.14 != 9.8? True

if __name__ == '__main__':
 main()
```

***

Karakter

Tipe data karakter adalah tipe data yang digunakan untuk memodelkan satu buah karakter. Tergantung implementasi dari bahasa pemrograman, satu buah karakter dapat berukuran satu byte (8 bit) atau satu word (16 bit). Bilamana karakter dimodelkan menggunakan satu byte, biasanya menggunakan pemetaan karakter ASCII (American Standard Code for Information Interchange). Sedangkan bilamana menggunakan satu word, menggunakan unicode.

Berbeda dengan ASCII yang hanya berisi kumpulan karakter latin, pada unicode karakter yang disimpan bervariasi. Karakter kanji cina, karakter hiragana/katakana jepang, karakter hangul korea, karakter arab, dan karakter pada bahasa lainnya termasuk dalam unicode. Jadi bilamana sebuah aplikasi dapat digunakan multi-bahasa maka dibutuhkan pengkodean unicode.

Bahasa pemrograman C#, JAVA, dan Python mengakomodir kedua pengkodean ini. Ketiga bahasa ini menggunakan tanda petik tunggal untuk menuliskan sebuah konstanta karakter. Ketiga bahasa ini juga memiliki kesamaan dalam operator relasional yang digunakan pada tipe data karakter.

Operator	Arti	Contoh	Hasil
<	Lebih kecil	'a' < 'c'	true
<=	Lebih kecil atau sama dengan	'a' <= 'c'	true
>	Lebih besar	'a' > 'c'	false
>=	Lebih besar atau sama dengan	'a' >= 'c'	false
==	Sama dengan	'a' == 'c'	false
!=	Tidak sama dengan	'a' != 'c'	true

Berikut ini adalah contoh penggunaan operator relasional menggunakan bahasa pemrograman C#.

```
using System;

class Program
{
 static void Main(string[] args)
 {
 char c1 = 'a', c2 = 'c';

 Console.WriteLine("{0} < {1}? {2}", c1, c2, (c1 < c2));
 Console.WriteLine("{0} <= {1}? {2}", c1, c2, (c1 <= c2));
 Console.WriteLine("{0} > {1}? {2}", c1, c2, (c1 > c2));
 Console.WriteLine("{0} >= {1}? {2}", c1, c2, (c1 >= c2));
 Console.WriteLine("{0} == {1}? {2}", c1, c2, (c1 == c2));
 Console.WriteLine("{0} != {1}? {2}", c1, c2, (c1 != c2));

 // Keluaran Program
 // a < c? True
 // a <= c? True
 // a > c? False
 // a >= c? False
 // a == c? False
 // a != c? True
 }
```

```
}
```

Sedangkan berikut ini adalah contoh serupa dalam bahasa pemrograman JAVA.

```
public class Program {
 public static void main(String[] args) {
 char c1 = 'a', c2 = 'c';

 System.out.println(c1 + " < " + c2 + "? " + (c1 < c2));
 System.out.println(c1 + " <= " + c2 + "? " + (c1 <= c2));
 System.out.println(c1 + " > " + c2 + "? " + (c1 > c2));
 System.out.println(c1 + " >= " + c2 + "? " + (c1 >= c2));
 System.out.println(c1 + " == " + c2 + "? " + (c1 == c2));
 System.out.println(c1 + " != " + c2 + "? " + (c1 != c2));

 // Keluaran Program
 // a < c? True
 // a <= c? True
 // a > c? False
 // a >= c? False
 // a == c? False
 // a != c? True
 }
}
```

Berikut adalah contoh dalam bahasa Python.

```
def main():
 c1 = 'a'
 c2 = 'c'

 print(c1 , "<" , c2 , "?" , (c1 < c2))
 print(c1 , "<=" , c2 , "?" , (c1 <= c2))
 print(c1 , ">" , c2 , "?" , (c1 > c2))
 print(c1 , ">=" , c2 , "?" , (c1 >= c2))
 print(c1 , "==" , c2 , "?" , (c1 == c2))
 print(c1 , "!=" , c2 , "?" , (c1 != c2))

 # Keluaran Program
 # a < c? True
 # a <= c? True
 # a > c? False
 # a >= c? False
 # a == c? False
 # a != c? True

if __name__ == '__main__':
 main()
```

***

String

Tipe data string adalah tipe data yang digunakan untuk memodelkan sebuah teks (kumpulan karakter). Berbeda dengan tipe data karakter yang menggunakan petik tunggal, tipe data string menggunakan tanda petik ganda untuk menuliskan konstanta teks.

Implementasi tipe data string pada bahasa pemrograman C#, JAVA, dan Python sedikit berbeda satu dengan yang lain. Hal ini mengakibatkan operator yang dapat digunakan untuk tiap bahasa, berbeda satu dengan yang lainnya.

Bahasa pemrograman C# memiliki sebuah operator untuk menggabungkan string (+) dan beberapa operator relasional untuk membandingkan satu string dengan string lainnya. Selain itu bahasa pemrograman ini juga memiliki banyak method yang digunakan untuk mengolah string itu sendiri. Berikut adalah contoh program yang mendemonstrasikan penggunaan tipe data string beserta operatornya.

```
using System;

class Program
{
 static void Main(string[] args)
 {
 string s1 = "hello";
 string s2 = "ello";
 string s3 = "h" + s2;

 Console.WriteLine("{0} + {1} = {2}", "h", s2, s3);
 Console.WriteLine("{0} == {1}? {2}", s1, s3, (s1 == s3));
 Console.WriteLine("{0} != {1}? {2}", s1, s2, (s1 != s2));

 // Keluaran Program
 // h + ello = hello
 // hello == hello? True
 // hello != ello? True
 }
}
```

Bahasa pemrograman JAVA memiliki pendekatan yang sedikit berbeda dengan bahasa pemrograman C#. Pada bahasa pemrograman JAVA operasi penggabungan string menggunakan operator (+), sedangkan relasional menggunakan pemanggilan method. Berikut ini adalah contoh program dalam bahasa pemrograman JAVA (serupa dengan bahasa pemrograman C#).

```
public class Program {
 public static void main(String[] args) {
 String s1 = "hello";
 String s2 = "ello";
 String s3 = "h" + s2;

 System.out.println("h" + " + " + s2 + " = " + s3);
 System.out.println(s1 + " == " + s3 + "? " + s1.equals(s3));
 System.out.println(s1 + " != " + s2 + "? " + !s1.equals(s2));

 // Keluaran Program
 // h + ello = hello
 // hello == hello? True
 // hello != ello? True
```

```
 }
}
```

Pada Python operator (+) digunakan sebagai penggabungan untuk String. Selain operator (+), terdapat dua operator lain untuk String, yaitu (*) dan in.

Operator	Arti	Contoh	Hasil
+	Penggabungan	"sa" + "ya"	"saya"
*	Pengulangan	'a' * 3	"aaa"
in	Kemunculan string dalam string	"a" in "saya"	true

Berikut ini adalah contoh program penggunaan tipe data String dalam bahasa pemrograman Python.

```
def main():
 s1 = "hello"
 s2 = "ello"
 s3 = "h" + s2

 print("h +" , s2 , "=" , s3)
 print(s1 , "==" , s3 , "?" , s1 == s3)
 print(s1 , "!=" , s2 , "?" , not s1 == s2)

 # Keluaran Program
 # h + ello = hello
 # hello == hello? True
 # hello != ello? True

if __name__ == '__main__':
 main()
```

***

Analisis Kasus

Pada penyelesaian program sering juga dihadapkan dengan keadaan dimana harus melakukan pengecekan terhadap sebuah kondisi sebelum melakukan sesuatu. Pengecekan kondisi ini dapat terjadi ketika melakukan validasi data yang dimasukkan oleh pengguna, pengecekan pemenuhan syarat, pengecekan range dua buah nilai, dan lain sebagainya. Untuk mengatasi hal tersebut, dalam Bahasa pemrograman pun disediakan perintah untuk melakukan pengecekan terhadap kondisi yang diperlukan.

Sebelum membuat program, lakukan analisis terlebih dahulu terhadap kasus yang akan diselesaikan. Pisahkan kondisi apa saja yang harus diperiksa dan aksi apa yang akan dijalankan ketika setiap kondisi terpenuhi.

Contohnya, kasus yang dihadapi adalah menentukan wujud zat berdasarkan suhunya. Pada suhu dibawah 0 derajat celcius wujud zat yang akan dihasilkan adalah padat (beku). Pada suhu antara 0 derajat celcius hingga dibawah 100 derajat celcius wujud zat adalah cair. Sedangkan dengan suhu diatas 100 derajat celcius, maka suhu akan menguap menjadi gas (uap).

Hasil analisis pemisahan kondisi dan aksi yang akan dilakukan:

Langkah	Kondisi	Aksi
1.	Suhu < 0	Wujud = "Padat"
2.	0 <= Suhu < 100	Wujud = "Cair"
3.	Suhu >= 100	Wujud = "Gas"

***

Masukan (Input) Pengguna

Pada bab sebelumnya telah dibahas berbagai cara untuk menampilkan data ke layar. Pada sub bab ini akan dibahas mengenai cara menerima masukan dari pengguna. Khusus untuk masukan tiap bahasa pemrograman memiliki cara yang sedikit berbeda satu dengan yang lainnya.

Pada aplikasi konsol, bahasa pemrograman C# menggunakan method Console.ReadLine() untuk menerima masukan (string) dari pengguna. Pada bahasa pemrograman C# hampir sebagian besar metode input menghasilkan sebuah tipe data string. Untuk mendapatkan tipe data lainnya perlu melakukan proses parsing terlebih dahulu. Proses parsing adalah sebuah proses menerjemahkan sebuah data menjadi bentuk lainnya sehingga mudah untuk diolah. Dalam kasus ini mengubah tipe data string menjadi tipe data numerik. Berikut ini adalah potongan kode program C# yang melakukan proses input, sampai dengan proses parsing nilai ke bilangan bulat (int).

```
Console.Write("Masukan sebuah nilai: ");
string nilaiStr = Console.ReadLine();

// Parsing string ke tipe int
int nilai;
int.TryParse(nilaiStr, out nilai);
```

Bahasa pemrograman JAVA menggunakan pendekatan yang berbeda dengan bahasa pemrograman C#. Pada bahasa pemrograman JAVA, untuk menerima masukan menggunakan class Scanner. Class ini digunakan untuk melakukan parsing nilai ke tipe data umum yang ada pada bahasa pemrograman JAVA. Method Scanner.next() digunakan untuk mengambil nilai dengan tipe data yang diinginkan (contoh bilamana ingin mengambil sebuah int, maka dapat menggunakan method Scanner.nextInt()). Berikut ini adalah potongan kode program JAVA yang melakukan proses input sebuah nilai bilangan bulat (int).

```
// Class Scanner digunakan untuk parsing masukan
Scanner scanner = new Scanner(System.in);

System.out.print("Masukan sebuah nilai: ");
int nilai;
nilai = scanner.NextInt();
```

Bahasa pemrograman Python memiliki pendekatan yang jauh lebih sederhana daripada bahasa pemrograman C# dan JAVA. Menerima masukan pada bahasa pemrograman Python menggunakan fungsi input(). Sedangkan proses parsing dapat langsung menggunakan tipe data yang bersangkutan, contoh untuk mendapatkan sebuah int maka dapat menggunakan int(input()). Berikut ini adalah potongan kode program Python yang melakukan proses input yang serupa dengan bahasa pemrograman C# dan JAVA.

```
nilai = int(input("Masukan sebuah nilai:");
```

***

Pernyataan if

Pada dasarnya konsep dari pernyataan if untuk seluruh bahasa pemrograman sama, akan tetapi ada sedikit perbedaan pada sintaks/kode perintah pada setiap bahasa pemrograman. Pernyataan if nantinya akan berpasangan dengan else if dan else, dimana ketika kondisi pada if tidak terpenuhi, maka program akan mengecek kondisi pada else if atau else. Sehingga hanya akan ada satu aksi saja yang dijalankan oleh program. Aksi pada else akan dijalankan ketika seluruh kondisi pada if dan else if tidak ada yang terpenuhi.

Berikut adalah tabel perbandingan pernyataan if di tiap bahasa pemrograman.

Umum	C#/JAVA	Python

if(kondisi1)
then aksi1
else if(kondisi2)
then aksi2
else if(kondisi3)
then aksi3
else
then aksi4
	if(kondisi1)
	{
	// aksi1
	}
	else if(kondisi2)
	{
	// aksi2
	}
	else if(kondisi3)
	{
	// aksi3
	}
	else
	{
	// aksi4
	}
		if(kondisi1):
		# aksi1
		elif(kondisi2):
		# aksi2
		elif(kondisi3):
		# aksi3
		else
		# aksi4

Analisis kasus yang biasanya memiliki tiga buah varian, analisis kasus tunggal (analisis satu kasus), analisis kasus ganda (analisis dua kasus), dan analisis banyak kasus. Analisis kasus tunggal: Buatlah sebuah kode program untuk mendeteksi apakah nilai yang dimasukan adalah sebuah bilangan ganjil. Berikut ini adalah solusi dalam bahasa pemrograman C#.

```
using System;

class Program
{
 static void Main(string[] args)
 {
 Console.Write("Masukan sebuah nilai: ");
 string nilaiStr = Console.ReadLine();

 int nilai;
 int.TryParse(nilaiStr, out nilai);

 if(nilai % 2 == 1)
 {
 Console.WriteLine("Nilai {0} ganjil.", nilai);
 }
 }
}
```

Berikut adalah solusi dalam bahasa pemrograman JAVA.

```
import java.util.Scanner;

public class Program {
 public static void main(String[] args) {
 Scanner sc = new Scanner(System.in);

 System.out.print("Masukan sebuah nilai: ");
 int nilai = sc.nextInt();

 if(nilai % 2 == 1) {
 System.out.println("Nilai " + nilai + " ganjil.");
 }
 }
}
```

Sedangkan berikut adalah solusi dalam bahasa Python.

```
def main():
 nilai = int(input("Masukan sebuah nilai:"))

 if(nilai % 2 == 1):
 print("Nilai", nilai, "ganjil.")

if __name__ == '__main__':
 main()
```

nilai	Evaluasi	Output
25	25 % 2 == 1, True	Nilai 25 ganjil.
30	30 % 2 == 1, False	-

Analisis kasus ganda: Buatlah sebuah kode program untuk mendeteksi apakah nilai yang dimasukan sebuah bilangan ganjil atau bilangan genap. Berikut adalah solusi dalam bahasa pemrograman C#.

```
using System;

class Program
{
 static void Main(string[] args)
 {
 Console.Write("Masukan sebuah nilai: ");
 string nilaiStr = Console.ReadLine();

 int nilai;
 int.TryParse(nilaiStr, out nilai);

 if(nilai % 2 == 1)
 {
 Console.WriteLine("Nilai {0} ganjil.", nilai);
 }
 else
 {
 Console.WriteLine("Nilai {0} genap.", nilai);
 }
 }
}
```

Berikut adalah solusi dalam bahasa pemrograman JAVA.

```
import java.util.Scanner;

public class Program {
 public static void main(String[] args) {
 Scanner sc = new Scanner(System.in);

 System.out.print("Masukan sebuah nilai: ");
 int nilai = sc.nextInt();

 if(nilai % 2 == 1) {
 System.out.println("Nilai " + nilai + " ganjil.");
 }
 else {
 System.out.println("Nilai " + nilai + " genap.");
 }
 }
}
```

Sedangkan berikut adalah solusi dalam bahasa Python.

```
def main():
 nilai = int(input("Masukan sebuah nilai:"))

 if(nilai % 2 == 1):
 print("Nilai", nilai, "ganjil.")
 else:
 print("Nilai", nilai, "genap.")

if __name__ == '__main__':
 main()
```

nilai	Evaluasi	Output
25	25 % 2 == 1, True	Nilai 25 ganjil.
30	30 % 2 == 1, False (kode blok else dijalankan)	Nilai 30 genap.

Analisis banyak kasus: Buatlah sebuah kode program untuk menampilkan wujud dari air dari suhu (dalam celcius) yang dimasukan. Berikut ini adalah solusi dalam bahasa pemrograman C#.

```
using System;

class Program
{
 static void Main(string[] args)
 {
 Console.Write("Masukan suhu air: ");
 string suhuStr = Console.ReadLine();

 double suhu;
 double.TryParse(suhuStr, out suhu);

 if(suhu < 0)
 {
 Console.WriteLine("Wujud padat.");
```

```
 }
 else if(suhu < 100)
 {
 Console.WriteLine("Wujud cair.");
 }
 else
 {
 Console.WriteLine("Wujud gas.");
 }
 }
}
```

Berikut adalah solusi serupa dalam bahasa pemrograman JAVA.

```
import java.util.Scanner;

public class Program {
 public static void main(String[] args) {
 Scanner sc = new Scanner(System.in);

 System.out.print("Masukan suhu air: ");
 double suhu = sc.nextDouble();

 if(suhu < 0) {
 System.out.println("Wujud padat.");
 }
 else if(suhu < 100) {
 System.out.println("Wujud cair.");
 }
 else{
 System.out.println("Wujud gas.");
 }
 }
}
```

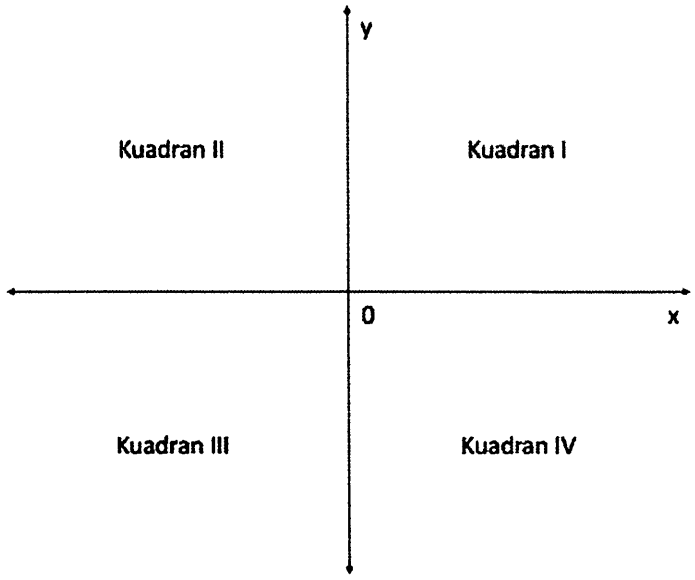
Berikut adalah solusi dalam bahasa pemrograman Python.

```
def main():
 suhu = float(input("Masukan suhu air:"))

 if(suhu < 0):
 print("Wujud padat.")
 elif(suhu < 100):
 print("Wujud cair.")
 else:
 print("Wujud gas.")

if __name__ == '__main__':
 main()
```

suhu	Evaluasi	Output
-20	suhu < 0, True	Wujud padat.
40	suhu < 0, False suhu < 100, True	Wujud cair.
121	suhu < 0, False	Wujud gas.

	suhu < 100, False (kode blok else dijalankan)	

Perlu diingat kondisi yang diperiksa oleh pernyataan if dan else if dapat lebih dari satu kondisi. Jika terdapat lebih dari sebuah kondisi yang akan diperiksa maka dapat menggunakan operator AND dan OR. Contoh kasus: Buatlah sebuah program yang akan memeriksa letak titik koordinat x dan y, sehingga dapat diketahui kuadran dari titik koordinat tersebut.



Kuadran	Kondisi
Titik pusat	X = 0, Y = 0
I	X > 0, Y > 0
II	X < 0, Y > 0
III	X < 0, Y < 0
IV	X > 0, Y < 0

Berikut adalah solusi dalam bahasa pemrograman C#.

```
using System;

class Program
{
 static void Main(string[] args)
 {
 Console.Write("X = ");
 string xStr = Console.ReadLine();
 Console.Write("Y = ");
 string yStr = Console.ReadLine();

 int x, y;
```

```
 int.TryParse(xStr, out x);
 int.TryParse(yStr, out y);

 if(x > 0 && y > 0)
 Console.WriteLine("Titik ({0}, {1}) berada di Kuadran I.", x, y);
 else if(x < 0 && y > 0)
 Console.WriteLine("Titik ({0}, {1}) berada di Kuadran II.", x, y);
 else if(x < 0 && y < 0)
 Console.WriteLine("Titik ({0}, {1}) berada di Kuadran III.", x, y);
 else if(x > 0 && y < 0)
 Console.WriteLine("Titik ({0}, {1}) berada di Kuadran IV.", x, y);
 else
 Console.WriteLine("Titik ({0}, {1}) berada di titik pusat.", x, y);
 }
}
```

Berikut adalah solusi dalam bahasa JAVA.

```
import java.util.Scanner;

public class Program {
 public static void main(String[] args) {
 Scanner sc = new Scanner(System.in);

 System.out.print("X = ");
 int x = sc.nextInt();
 System.out.print("Y = ");
 int y = sc.nextInt();

 if(x > 0 && y > 0)
 System.out.println("Titik (" + x + ", " + y + ") berada di Kuadran I.");
 else if(x < 0 && y > 0)
 System.out.println("Titik (" + x + ", " + y + ") berada di Kuadran II.");
 else if(x < 0 && y < 0)
 System.out.println("Titik (" + x + ", " + y + ") berada di Kuadran III.");
 else if(x > 0 && y < 0)
 System.out.println("Titik (" + x + ", " + y + ") berada di Kuadran IV.");
 else
 System.out.println("Titik (" + x + ", " + y + ") berada di titik pusat.");
 }
}
```

Sedangkan berikut adalah solusi dalam bahasa Python.

```
def main():
 x = int(input("X ="))
 y = int(input("Y ="))

 if(x > 0 and y > 0):
 print("Titik (", x, ",", y, ") berada di Kuadran I.", sep = "")
 elif(x < 0 and y > 0):
 print("Titik (", x, ",", y, ") berada di Kuadran II.", sep = "")
 elif(x < 0 and y < 0):
 print("Titik (", x, ",", y, ") berada di Kuadran III.", sep = "")
 elif(x > 0 and y < 0):
 print("Titik (", x, ",", y, ") berada di Kuadran IV.", sep = "")
 else:
 print("Titik (", x, ",", y, ") berada di titik pusat.", sep = "")

if __name__ == '__main__':
 main()
```

Pernyataan if Bersarang

Pernyataan if dapat digunakan juga untuk kasus-kasus yang memerlukan pemeriksaan kondisi secara berlapis. Pernyataan if dapat digunakan di dalam pernyataan if lainnya, sehingga dapat melakukan pemeriksaan kondisi secara berlapis. Hal ini dikenal dengan istilah pernyataan if bersarang. Contoh kasus: Buatlah sebuah program yang menerima sebuah masukan nilai kemudian menampilkan apakah nilai tersebut adalah ganjil positif, ganjil negatif, genap positif, atau genap negatif.

Berikut ini adalah solusi dalam bahasa C#.

```
using System;

class Program
{
 static void Main(string[] args)
 {
 Console.Write("Input sebuah nilai: ");
 string nilaiStr = Console.ReadLine();

 int nilai;
 int.TryParse(nilaiStr, out nilai);

 if(nilai % 2 != 0)
 { // Ganjil
 if(nilai > 0)
 Console.WriteLine("Ganjil Positif.");
 else
 Console.WriteLine("Ganjil Negatif.");
 }
 else
 { // Genap
 if(nilai > 0)
 Console.WriteLine("Genap Positif.");
 else
 Console.WriteLine("Genap Negatif.");
 }
 }
}
```

Sedangkan berikut adalah solusi dalam bahasa JAVA.

```
import java.util.Scanner;

public class Program {
 public static void main(String[] args) {
 Scanner sc=new Scanner(System.in);

 System.out.print("Input sebuah nilai: ");
 int nilai = sc.nextInt();

 if(nilai % 2 != 0) { // Ganjil
 if(nilai > 0)
 System.out.print("Ganjil Positif.");
 else
 System.out.print("Ganjil Negatif.");
 }
 else { // Genap
 if(nilai > 0)
 System.out.print("Genap Positif.");
```

```
 else
 System.out.print("Genap Negatif.");
 }
 }
}
```

Berikut adalah solusi dalam Python.

```
def main():
 nilai = int(input("Input sebuah nilai:"))

 if(nilai % 2 != 0): # Ganjil
 if(nilai > 0):
 print("Ganjil Positif.")
 else:
 print("Ganjil Negatif.")
 else: # Genap
 if(nilai > 0):
 print("Genap Positif.")
 else:
 print("Genap Negatif.")

if __name__ == '__main__':
 main()
```

***

Pengulangan

Pada pemrograman terdapat juga perintah untuk pengulangan. Perintah ini digunakan ketika satu paket perintah akan dijalakankan berkali-kali. Pada setiap algoritma, pengulangan memiliki sebuah kondisi yang akan menyebabkan pengulangan tersebut berhenti. Pengulangan terdiri dari dua bagian besar, yaitu evaluasi kondisi pengulangan dan kumpulan aksi yang ingin diulang.

Evaluasi kondisi pengulangan digunakan untuk mengevaluasi syarat kumpulan kode program untuk diulang. Kondisi ini dibuat dalam sebuah pernyataan boolean yang akan menghasilkan nilai benar/salah (true/false). Selama kondisi dari pengulangan benar (true), maka kumpulan aksi yang ada pada pengulangan akan terus menerus dijalankan.

Sedangkan kumpulan aksi berisi satu atau lebih aksi yang ingin dijalankan berulang oleh blok pengulangan. Kumpulan aksi ini akan terus dijalankan selama kondisi pengulangan bernilai benar (true) atau sampai kondisi pengulangan bernilai salah (false).

Contoh bilamana Anda diminta untuk membuat sebuah program untuk menghitung rata-rata nilai dari 40 orang siswa yang dimasukan pengguna, maka program akan melakukan pengulangan input nilai sebanyak 40 kali. Setelah didapat data nilai dari 40 siswa, maka program akan melakukan pengulangan penjumlahan nilai sebanyak 40 kali, lalu jumlahnya dibagi 40 untuk mendapatkan nilai rata-ratanya.

***

Keluaran (Output)

Sebelum masuk ke pembahasan lebih rinci mengenai pengulangan ada baiknya kali ini melakukan pembahasan mengenai mencetak keluaran. Pada bab sebelumnya (bilamana dilihat secara rinci), beberapa bahasa pemrograman memiliki setidaknya dua buah varian fungsi atau method untuk menampilkan data ke layar konsol.

Pada bahasa C# terdapat Console.Write() dan Console.WriteLine(). Perbedaannya adalah pada Console.Write() hanya mencetak hasil ke layar console, sedangkan pada Console.WriteLine() setelah mencetak hasil lalu dilanjutkan dengan perintah untuk pindah ke baris baru.

```
using System;

class Program
{
 static void Main(string[] args)
 {
 // Hasil Keluaran
 // Hello
 // World
 Console.WriteLine("Hello");
 Console.WriteLine("World");

 // Hasil Keluaran
 // HelloWorld
 Console.Write("Hello");
 Console.WriteLine("World");
 }
}
```

Pada bahasa JAVA terdapat System.out.print() dan System.out.println() yang kegunaan serupa dengan kedua fungsi/method milik bahasa pemrograman C#. System.out.print() digunakan untuk menampilkan hasil, sedangkan System.out.println() digunakan untuk menampilkan hasil dan pindah ke baris baru.

```
public class Program {
 public static void main(String[] args) {
 // Hasil Keluaran
 // Hello
 // World
 System.out.println("Hello");
 System.out.println("World");

 // Hasil Keluaran
 // HelloWorld
 System.out.print("Hello");
 System.out.println("World");
 }
}
```

Berbeda sedikit dengan bahasa C# dan bahasa JAVA, pada bahasa Python hanya mengenal satu buah saja fungsi untuk mencetak ke layar konsol yaitu print(). Fungsionalitas yang sama dapat dilakukan oleh bahasa Python dengan menambahkan nilai pada parameter di fungsi print() itu sendiri. Parameter end, digunakan untuk mengubah penutup dari fungsi print(). Parameter end memiliki nilai default pindah baris.

```
def main():
 # Hasil Keluaran
 # Hello
 # World
 print("Hello")
 print("World")

 # Hasil Keluaran
 # Hello World
 print("Hello", end = " ")
 print("World")

if __name__ == '__main__':
 main()
```

Sedangkan parameter sep, digunakan untuk mengubah pemisah antara satu nilai ke nilai lainnya. Parameter sep memiliki nilai default sebuah spasi.

```
def main():
 # Hasil Keluaran
 # Hello World
 print("Hello", "World")

 # Hasil Keluaran
 # Hello*World
 print("Hello", "World", sep = "*")

if __name__ == '__main__':
 main()
```

***

Pernyataan while-do

Pernyataan while-do cocok digunakan untuk melakukan pengulangan yang jumlah pengulangan belum diketahui di awal. Contoh ketika sebuah program diminta untuk terus menerus menerima masukan dari pengguna sampai pengguna memasukan "done". Program tidak pernah mengetahui berapa kali proses input harus dilakukan, karena jumlahnya tergantung dari pengguna itu sendiri. Berikut ini adalah format umum pernyataan while-do.

```
[inisialisasi]
while(kondisi) do
 aksi
 [peubah kondisi]
```

Bagian inisialisasi digunakan untuk memberikan nilai awal pada variabel yang digunakan pada bagian evaluasi kondisi sehingga pengulangan dapat berjalan dengan baik. Bagian kondisi digunakan untuk melakukan evaluasi nilai sebelum melakukan aksi. Aksi dilakukan bilamana kondisi masih menghasilkan nilai true. Peubah kondisi digunakan untuk mengubah nilai, sehingga pengulangan dimungkinkan untuk selesai (berhenti). Tergantung kasusnya, pada bagian peubah kondisi biasanya melakukan penambahan atau pengurangan nilai penghitung (counter). Berikut ini adalah contoh program C# yang bertujuan untuk menampilkan teks Hello sebanyak lima kali.

```
using System;

class Program
{
 static void Main(string[] args)
 {
 int counter = 1;
 while(counter <= 5)
 {
 Console.Write("Hello ");
 counter++;
 }
 }
}
```

Berikut ini adalah kode program JAVA yang serupa dengan kode program C# di atas.

```
public class Program {
 public static void main(String[] args) {
 int counter = 1;
 while(counter <= 5)
 {
 System.out.print("Hello ");
 counter++;
 }
 }
}
```

Sedangkan berikut adalah kode program Python.

```
def main():
 counter = 1
 while(counter <= 5):
 print("Hello", end = " ")
```

```
 counter = counter + 1

if __name__ == '__main__':
 main()
```

counter	Evaluasi	Output
1	1 <= 5, True	Hello
2	2 <= 5, True	Hello Hello
3	3 <= 5, True	Hello Hello Hello
4	4 <= 5, True	Hello Hello Hello Hello
5	5 <= 5, True	Hello Hello Hello Hello Hello
6	6 <= 5, False	(berhenti)

Pada kasus kedua akan mendemonstrasikan penggunaan pernyataan while-do untuk melakukan pengulangan yang jumlahnya belum diketahui. Pada kasus kali ini Program akan meminta pengguna untuk memasukan nama peserta dan program menghitung jumlah peserta yang dimasukan pengguna. Program akan berhenti meminta masukan nama bilamana dimasukan teks "DONE". Berikut adalah solusi dalam bahasa C#.

```
using System;

class Program
{
 static void Main(string[] args)
 {
 Console.WriteLine("Input nama peserta");

 Console.Write("Nama: ");
 string nama = Console.ReadLine();

 int count = 0;
 while(nama != "DONE")
 {
 count++;
 nama = Console.ReadLine();
 }

 Console.WriteLine("Jumlah peserta adalah {0} orang.", count);

 // Hasil Keluaran:
 // Input nama peserta
 // Nama: Budi
 // Nama: Iwan
 // Nama: Wati
 // Nama: DONE
 // Jumlah peserta adalah 3 orang.
 }
}
```

Berikut solusi serupa dalam bahasa JAVA.

```
import java.util.Scanner;

public class Program {
 public static void main(String[] args) {
 Scanner sc=new Scanner(System.in);

 System.out.println("Input nama peserta");

 System.out.print("Nama: ");
 String nama = sc.nextLine();

 int count = 0;
 while(!nama.equals("DONE")) {
 count++;
 nama = sc.nextLine();
 }

 System.out.println("Jumlah peserta adalah " + count + " orang.");

 // Hasil Keluaran:
 // Input nama peserta
 // Nama: Budi
 // Nama: Iwan
 // Nama: Wati
 // Nama: DONE
 // Jumlah peserta adalah 3 orang.
 }
}
```

Berikut adalah kode program dalam bahasa Python.

```
def main():
 print("Input nama peserta")

 count = 0
 nama = input("Nama:")

 while(nama != "DONE"):
 count = count + 1
 nama = input("Nama:")

 print("Jumlah peserta adalah", count,"orang.")

 # Hasil Keluaran:
 # Input nama peserta
 # Nama: Budi
 # Nama: Iwan
 # Nama: Wati
 # Nama: DONE
 # Jumlah peserta adalah 3 orang.

if __name__ == '__main__':
 main()
```

Pada bahasa pemrograman C# dan JAVA, selain pernyataan while-do ada juga pernyataan do-while. Perbedaannya adalah pada pernyataan do-while evaluasi kondisi dilakukan terakhir. Hal ini memungkinkan kode program dalam blok pengulangan dilakukan setidaknya sekali, walaupun kondisinya bernilai false. Berikut adalah format umum pernyataan do-while.

```
[inisialisasi]
```

```
do
 aksi
 [peubah kondisi]
while(kondisi)
```

Berikut ini adalah contoh kode program yang menggunakan pernyataan do-while dalam bahasa pemrograman C#.

```
using System;

class Program
{
 static void Main(string[] args)
 {
 Console.WriteLine("Pernyataan while-do");
 int value = 5;
 while(value <= 4)
 {
 Console.WriteLine(value);
 value++;
 }

 Console.WriteLine("Pernyataan do-while");
 value = 5;
 do
 {
 Console.WriteLine(value);
 value++;
 } while(value <= 4);

 // Hasil Keluaran:
 // Pernyataan while-do
 // Pernyataan do-while
 // 5
 }
}
```

Sedangkan berikut adalah kode program dalam bahasa JAVA.

```
public class Program {
 public static void main(String[] args) {
 System.out.println("Pernyataan while-do");
 int value = 5;
 while(value <= 4) {
 System.out.println(value);
 value++;
 }

 System.out.println("Pernyataan do-while");
 value = 5;
 do {
 System.out.println(value);
 value++;
 } while(value <= 4);

 // Hasil Keluaran:
 // Pernyataan while-do
 // Pernyataan do-while
 // 5
 }
}
```

Perhatikan bahwa ketika menggunakan pernyataan while-do blok kode pengulangan tidak dijalankan, karena kondisi bernilai false, sehingga pengulangan langsung dihentikan tanpa menjalankan blok kode program pengulangan. Pada bagian pernyataan do-while, walaupun menggunakan nilai yang sama tetapi kode program dijalankan sekali, karena bagian evaluasi dilakukan terakhir.

***

Pernyataan for

Berbeda dengan pernyataan while-do, pernyataan for digunakan bilamana jumlah pengulangan yang akan dilakukan sudah diketahui sebelum pengulangan tersebut dimulai. Sama dengan pernyataan while-do, pada pernyataan for ada bagian evaluasi kondisi untuk menentukan apakah pengulangan akan berlanjut atau berhenti.

Berbeda dengan pernyataan while-do yang memiliki format umum yang sama untuk ketiga bahasa pemrograman. Pada pernyataan for, bahasa pemrograman C# dan JAVA memiliki format yang sama, sedangkan bahasa pemrograman Python memiliki format yang berbeda. Berikut adalah format umum pernyataan for untuk C# dan JAVA:

```
for(inisialiasi;kondisi;peubah kondisi)
 aksi
```

Inisialiasi berisi kode program untuk memasukan nilai awal ke variabel counter (contoh: int i = 0). Kondisi berisi ekspresi boolean yang menentukan batas pengulangan. Selama ekspresi boolean pada bagian kondisi bernilai true, maka kode program aksi akan terus dijalankan (contoh: i < 5, selama nilai i masih lebih kecil dari 5, maka kode program aksi akan dijalankan). Peubah kondisi berisi pernyataan yang bertujuan merubah nilai variabel counter sehingga pengulangan dapat berhenti (kondisi tercapai, menghasilkan nilai false). Variabel counter sebaiknya menggunakan bilangan bulat, agar jumlah pengulangan tepat. Berikut ini adalah contoh program sederhana dalam bahasa C# yang bertujuan untuk menampilkan teks Hello sebanyak lima kali.

```
using System;

class Program
{
 static void Main(string[] args)
 {
 for(int i = 0; i < 5; ++i)
 Console.WriteLine("Hello");
 }
}
```

Sedangkan berikut adalah solusi dalam bahasa JAVA.

```
public class Program {
 public static void main(String[] args) {
 for(int i = 0; i < 5; ++i)
 System.out.println("Hello");
 }
}
```

Pada bahasa Python, format penulisan pernyataan for sedikit berbeda. Berikut adalah format pernyataan for dalam bahasa Python.

```
for varCounter in range([nilai awal], [nilai akhir], [langkah])
```

Berbeda dengan bahasa C# dan JAVA, bahasa Python melakukan iterasi nilai dari rentang nilai (range). Range akan membuat sebuah kumpulan nilai yang dimulai dari nilai awai, sampai nilai akhir, dengan besar langkah tertentu. Contoh: Pernyataan range(3) akan menghasilkan kumpulan nilai 0, 1, 2. Pernyataan range(1, 3) akan menghasilkan kumpulan 1, 2. Sedangkan pernyataan range(1, 20, 2) akan menghasilkan kumpulan 1, 3,

5, 7, 9, 11, 13, 15, 17, 19. Berikut adalah solusi dalam bahasa Python untuk menampilkan teks Hello sebanyak lima kali.

```
def main():
 for i in range(1, 5+1, 1):
 print("Hello", end = " ")

if __name__ == '__main__':
 main()
```

Bagaimana bilamana ingin melakukan pengulangan yang nilainya dari besar ke kecil? Berikut ini adalah contoh solusi sederhana menggunakan bahasa pemrograman C#.

```
using System;

class Program
{
 static void Main(string[] args)
 {
 for(int i = 20; i > 0; i -= 2)
 Console.WriteLine(i);
 }
}
```

Sedangkan berikut dalam bahasa JAVA.

```
public class Program {
 public static void main(String[] args) {
 for(int i = 20; i > 0; i -= 2)
 System.out.println(i);
 }
}
```

Terakhir bilamana menggunakan bahasa Python.

```
def main():
 for i in range(20, 0, -2):
 print(i)

if __name__ == '__main__':
 main()
```

***

Pengulangan Bersarang

Pengulangan bersarang merupakan pengulangan yang didalamnya terdapat pengulangan lain. Cara kerja dari pengulangan bersarang ini ialah pada setiap putaran pengulangan terluar, maka pengulangan yang ada didalam akan dimulai dari awal dan diselesaikan hingga akhir. Perhatikan contoh yang akan menunjukan proses kerja dari pengulangan bersarang berikut ini:

Variable i mewakili pengulangan yang terluar, sedangkan variable j mewakili pengulangan yang berada didalam. Berikut adalah contoh kode program menggunakan bahasa C#.

```
using System;

class Program
{
 static void Main(string[] args)
 {
 for(int i = 1; i > 4; ++i)
 for(int j = 1; j > 5; ++j)
 Console.WriteLine("i = " + i + " j = " + j);
 }
}
```

Sedangkan berikut adalah kode program menggunakan bahasa JAVA.

```
public class Program {
 public static void main(String[] args) {
 for(int i = 1; i > 4; ++i)
 for(int j = 1; j > 5; ++j)
 System.out.println("i = " + i + " j = " + j);

 }
}
```

Dan terakhir kode program serupa menggunakan bahasa Python.

```
def main():
 for i in range(1, 3+1, 1):
 for j in range(1, 4+1, 1):
 print("i = ", i, " j = ", j, sep = "")

if __name__ == '__main__':
 main()
```

***

Fungsi

Sebuah organisasi terbentuk dari orang-orang yang memiliki peranan dan tugas masing-masing dalam pencapaian sebuah tujuan. Seluruh orang yang berada dalam organisasi tersebut bekerjasama untuk dapat mencapai keberhasilan. Sama halnya dengan organisasi tersebut, sebuah program yang ditujukan untuk menyelesaikan permasalahan yang banyak atau rumit akan dibentuk dalam modul-modul dimana setiap modul memiliki peran dan tugas masing-masing. Modul tersebut akan diimplementasikan dalam program berupa Fungsi. Sebuah Fungsi akan memiliki sebuah tugas dan dapat mengeluarkan sebuah hasil yang diperlukan untuk menyelesaikan permasalahan. Manfaat dari pembagian modul tersebut adalah:

1. Program menjadi lebih terstruktur.
2. Lebih mudah dalam melakukan penyelusuran kesalahan (error tracking).
3. Mengurangi penulisan kode program yang berulang.

Terkadang sebelum sebuah fungsi bekerja, fungsi tersebut membutuhkan informasi pendukung untuk menyelesaikan tugasnya. Informasi tersebut disebut sebagai Parameter. Jumlah parameter yang dibutuhkan sebuah fungsi tidak dibatasi. Layaknya sebuah variable, parameter yang ada pada setiap fungsi memiliki tipe data sesuai dengan data yang akan ditampungnya. Parameter inilah yang nantinya akan digunakan oleh fungsi dalam menjalankan tugasnya. Pada bahasa pemrograman Python, pendefinisian tipe data parameter tidak diperlukan. Berikut adalah sintaks umum deklarasi fungsi:

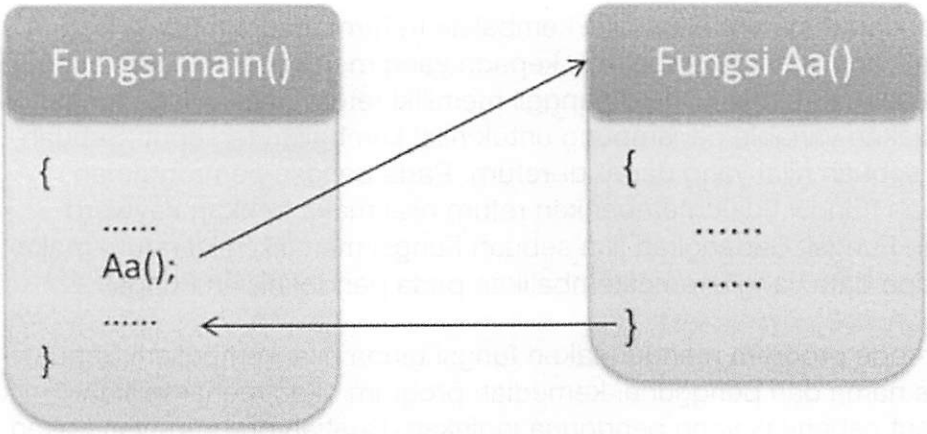
```
[tipe/keyword] namaFungsi(tipe param1, tipe param2, ...)
{
 aksi1
 aksi2
 ...
 aksiN

 return ...
}
```

Pada bahasa pemrograman C# dan JAVA pendefinisian fungsi diawali dengan tipe (contoh: int, float, dll) dari nilai yang dikembalikan (return). Sedangkan pada bahasa Python, diawali dengan keyword def.

Sebuah Fungsi akan menjalankan perintah didalamnya jika terdapat pemanggilan kepada fungsi tersebut. Pemanggilan fungsi dilakukan dengan cara menuliskan nama dari Fungsi yang akan dipanggil, diikuti dengan ( ). Pada saat pemanggilan fungsi ini, harus diperhatikan apakah fungsi yang dipanggil tersebut memiliki parameter atau tidak. Jika memiliki parameter, maka pada saat pemanggilan fungsi, pemanggil harus memberikan nilai untuk masing-masing parameter yang ada. Nilai yang akan diberikan untuk fungsi disertakan pada ( ) setelah penyebutan nama fungsi, sintaks:

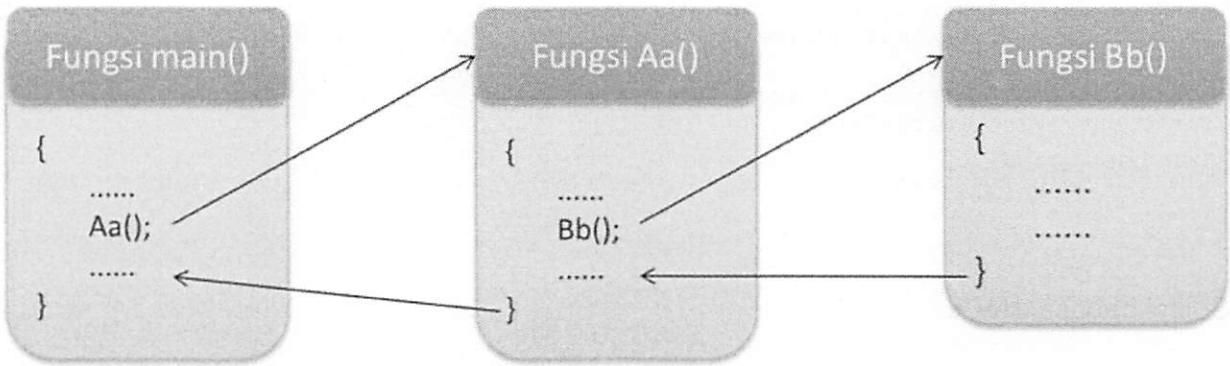
```
namaFungsi(nilai1,nilai2,...)
```

Jumlah nilai, tipe data, serta urutan dalam pemberian nilai harus sesuai dengan parameter pada fungsi. Nilai yang dikirimkan oleh pemanggil, akan ditampung pada setiap parameter yang ada sesuai dengan urutan pemberian nilai.



Gambar diatas menunjukan alur program dengan penggunaan fungsi. Pada saat program dijalankan maka compiler akan menjalankan perintah pada fungsi main() terlebih dahulu. Perintah pada fungsi main() akan dijalankan dari baris pertama sampai baris terakhir secara terurut. Jika pada deretan perintah tersebut terdapat perintah pemanggilan fungsi, contohnya memanggil fungsi Aa(), maka fungsi main() akan memanggil fungsi Aa() tersebut untuk kemudian menjalankan perintah-perintah yang ada pada fungsi Aa(). Setelah perintah pada fungsi Aa() selesai dijalankan, maka program akan kembali ke fungsi main() untuk menjalankan perintah pada baris berikutnya.

Pemanggilan fungsi bukan hanya dapat dilakukan oleh fungsi main() saja, melainkan dapat dilakukan oleh fungsi selain main(). Perhatikan proses pemanggilan antar fungsi pada gambar dibawah ini.



Fungsi main() akan selalu dijalankan pertama kali ketika program dijalankan. Pada alur program ini, terdapat pemanggilan fungsi Aa() pada fungsi main(). Maka seperti pada penjelasan sebelumnya, fungsi main() akan melakukan pemanggilan fungsi Aa() untuk kemudian program menjalankan fungsi Aa(). Pada fungsi Aa() perintah-perintah yang ada akan dijalankan secara beruturan mulai dari baris teratas. Diilustrasikan pada gambar di atas, bahwa pada fungsi Aa() terdapat pemanggilan kepada fungsi lain yaitu fungsi Bb(). Dikarenakan ada pemanggilan dari fungsi Aa(), maka perintah pada fungsi Bb() akan dijalankan mulai dari baris teratas sampai selesai. Setelah sampai pada baris paling akhir, maka program akan kembali pada fungsi Aa(). Jika pada fungsi Aa() masih terdapat perintah pada baris berikutnya, maka perintah tersebut akan diselesaikan terlebih dahulu sampai perintah paling akhir. Setelah perintah pada fungsi Aa() selesai dijalankan, maka program akan kembali pada fungsi main(). Program akan menjalankan perintah pada baris berikutnya setelah baris perintah pemanggilan fungsi Aa().

Setiap fungsi yang ada, dapat memberikan nilai kembalian (return) ataupun tidak. Return merupakan sebuah nilai yang akan dikembalikan kepada yang memanggil fungsi tersebut. Perlu diperhatikan, jika sebuah fungsi yang dipanggil memiliki return, maka pada fungsi pemanggil harus disediakan variable penampung untuk nilai kembalian tersebut. Sebuah Fungsi hanya memiliki sebuah nilai yang dapat di-return. Pada bahasa pemrograman Java dan C#, jika sebuah Fungsi tidak memberikan return nilai maka berikan keyword void pada pendefinisian Fungsi. Sedangkan jika sebuah Fungsi memiliki nilai return maka berikan pendefinisian tipe data yang akan dikembalikan pada pendefinisian Fungsi.

Berikut adalah contoh kode program menggunakan fungsi tanpa nilai kembalian. Sebuah program akan meminta nama dari pengguna, kemudian program akan menampilkan nama pengguna tersebut sebanyak yang pengguna inginkan. Buatlah program yang akan menampilkan nama pengguna tersebut.

Solusi bahasa C#.

```
using System;

class Program
{
 static void TampilNama(string pengguna, int jumlah)
 {
 for(int i = 1; i <= jumlah; ++i)
 Console.WriteLine(pengguna);
 }

 static void Main(string[] args)
 {
 Console.Write("Masukan nama Anda: ");
 string nama = Console.ReadLine();

 Console.Write("Berapa kali akan ditampilkan? ");
 int count = int.Parse(Console.ReadLine());

 TampilNama(nama, count);
 }
}
```

Solusi bahasa JAVA.

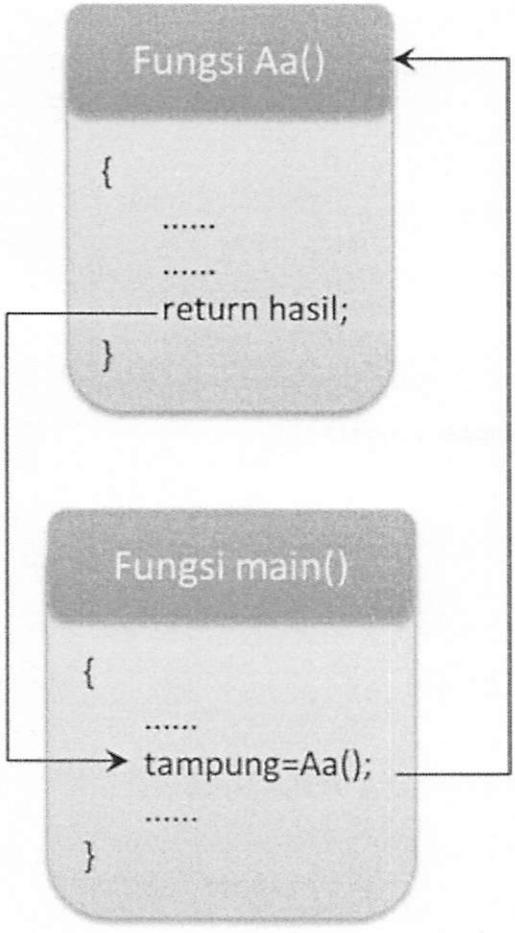
```
import java.util.Scanner;

public class Program {
 public static void tampilNama(String pengguna, int jumlah) {
 for(int i = 1; i <= jumlah; ++i)
 System.out.println(pengguna);
 }

 public static void main(String[] args) {
 Scanner sc = new Scanner(System.in);

 System.out.print("Masukan nama Anda: ");
 String nama = sc.next();

 System.out.print("Berapa kali akan ditampilkan? ");
 int count = sc.nextInt();

 tampilNama(nama, count);
 }
```

```
}
```

Solusi bahasa Python.

```
def tampilNama(pengguna, jumlah):
 for i in range(1, jumlah+1, 1):
 print(pengguna)

def main():
 nama = input("Masukan nama Anda: ")
 count = int(input("Berapa kali akan ditampilkan? "))
 tampilNama(nama, count)

if __name__ == '__main__':
 main()
```

Ilustrasi pembuatan fungsi dengan nilai kembalian.



Fungsi main() melakukan pemanggilan terhadap fungsi Aa(). Pada akhir fungsi terdapat perintah return dengan diikuti oleh sebuah variable bernama hasil. Nilai (value) dari variable hasil akan dikirimkan kepada fungsi main() dan diterima oleh variable tampung. Maka variable tampung nantinya akan memiliki nilai (value) hasil dari pemanggilan fungsi Aa().

Berikut adalah contoh kode program menggunakan fungsi dengan nilai kembalian. Sebuah program akan menghitung luas sebuah segitiga. Program akan meminta pengguna untuk memasukkan informasi berupa panjang alas dan tinggi dari segitiga.

Fungsi akan menghitung dan mengembalikan hasil perhitungan luas ke fungsi utama.

Solusi bahasa C#.

```
using System;

class Program
{
 static double HitungLuas(double alas, double tinggi)
 {
 double hasil = (alas * tinggi) / 2;
 return hasil;
 }

 static void Main(string[] args)
 {
 Console.Write("Alas: ");
 int alas = int.TryParse(Console.ReadLine());

 Console.Write("Tinggi: ");
 int tinggi = int.Parse(Console.ReadLine());

 double luas = HitungLuas(alas, tinggi);
 Console.WriteLine("Luas: " + luas);
 }
}
```

Solusi bahasa JAVA.

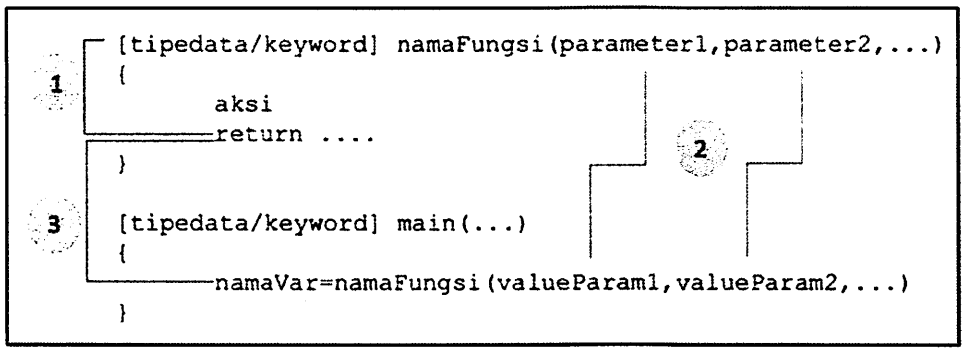
```
import java.util.Scanner;

public class Program {
 public static double hitungLuas(double alas, double tinggi) {
 double hasil = (alas * tinggi) / 2;
 return hasil;
 }

 public static void main(String[] args) {
 Scanner sc = new Scanner(System.in);

 System.out.print("Alas: ");
 int alas = sc.nextInt();

 System.out.print("Tinggi: ");
 int tinggi = sc.nextInt();

 double luas = hitungLuas(alas, tinggi);
 System.out.println("Luas: " + luas);
 }
}
```

Solusi bahasa Python.

```
def hitungLuas(alas, tinggi):
 hasil = (alas * tinggi) / 2
 return hasil

def main():
 alas = int(input("Alas: "))
 tinggi = int(input("Tinggi: "))
 luas = hitungLuas(alas, tinggi)
 print("Luas:", luas)
```

```
if __name__ == '__main__':
 main()
```

Setiap fungsi akan mulai bekerja jika ada sebuah pemanggilan terhadap fungsi tersebut. Ketika melakukan pemanggilan fungsi ada tiga hal yang harus diperhatikan:



1. Tipe data ketika mendefinisikan fungsi harus sama dengan tipe data nilai yang dikembalikan. Pada bahasa Python tipe data tersebut tidak perlu didefinisikan, cukup menggunakan keyword def.
2. Bilamana fungsi memiliki parameter, nilai untuk parameter tersebut harus diberikan pada saat pemanggilan fungsi. Perhatikan tipe data dan urutan dari parameternya.
3. Bilamana fungsi mengembalikan nilai (return) maka, pada fungsi utama (main) harus ada variable penampung nilai yang akan dikembalikan oleh fungsi atau dapat langsung ditampilkan hasilnya ke layar.

***

Fungsi Rekursif

Fungsi rekursif merupakan fungsi yang melakukan pemanggilan pada dirinya sendiri secara terus-menerus. Fungsi iniakan berhenti melakukan rekursif pada kondisi tertentu. Fungsi rekursif dibuat agar perintah yang dijalankan menjadi lebih sederhana dan efisien. Sebelum membuat Fungsi rekursif, ada beberapa hal yang harus diperhatikan terlebih dahulu:

1. Proses yang akan dijalankan berulang-ulang tetapi dengan nilai parameter yang dapat berubah-ubah.
2. Kondisi dimana Fungsi berhenti melakukan rekursif.

Contoh kasus 1, buatlah program untuk menghitung hasil perpangkatan sebuah bilangan. Sebuah bilangan a akan dipangkatkan dengan b.

Misal a = 2 (basis) dan b = 3 (eksponen) maka, proses dari perpangkatan secara matematika adalah:

$2^3 = 2 \times 2 \times 2$

Atau dapat dituliskan juga:

$$2^3 = 2 \times 2^2$$
$$\downarrow$$
$$2^2 = 2 \times 2^1$$
$$\downarrow$$
$$2^1 = 2 \times 2^0$$
$$\downarrow$$
$$2^0 = 1$$

Pada kasus ini, proses yang dilakukan berulang adalah proses perkalian dengan kondisi berhenti ketika pangkat adalah bilangan nol. Proses yang akan dijalankan pada Fungsi rekursif adalah perkalian a x a^b dimana nilai b akan selalu dikurangi 1 pada setiap prosesnya, dan proses akan berhenti ketika b = 0, dimana Fungsi akan mengembalikan nilai 1.

Solusi dalam bahasa C#.

```
using System;

class Program
{
 static int HitungPangkat(int basis, int eksponen)
 {
 if(eksponen == 0)
 return 1;
 else
 return basis * HitungPangkat(basis, eksponen-1);
 }
```

```
 static void Main(string[] args)
 {
 Console.Write("Basis: ");
 int a = int.TryParse(Console.ReadLine());

 Console.Write("Eksponen: ");
 int b = int.Parse(Console.ReadLine());

 int hasil = HitungPangkat(a, b);
 Console.WriteLine("Hasil: " + hasil);
 }
}
```

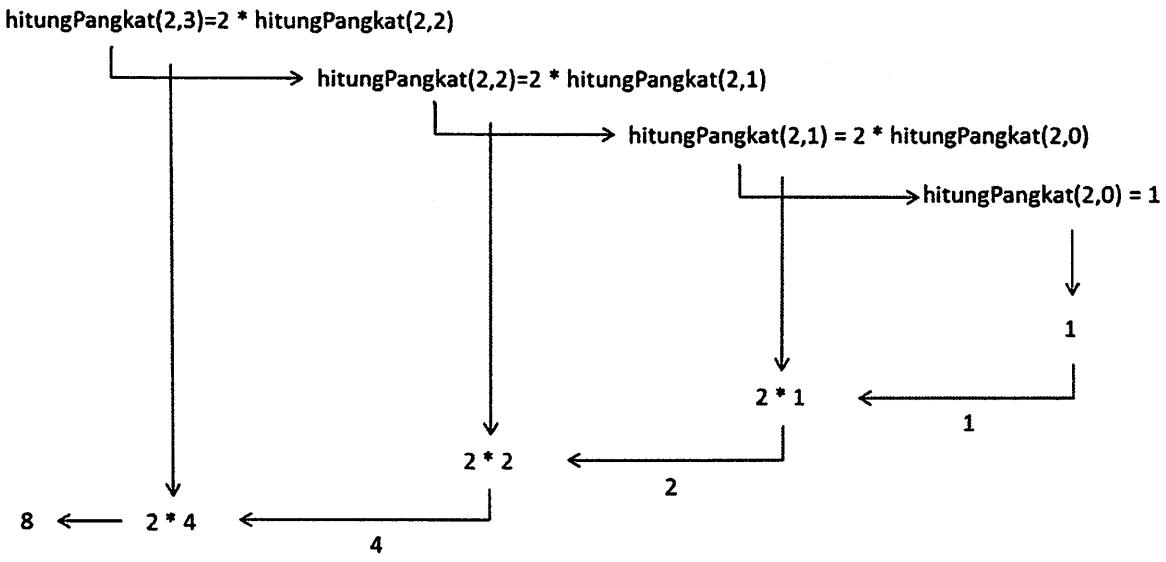
Solusi dalam bahasa JAVA.

```
import java.util.Scanner;

public class Program {
 public static int hitungPangkat(int basis, int eksponen) {
 if(eksponen == 0)
 return 1;
 else
 return basis * hitungPangkat(basis, eksponen-1);
 }

 public static void main(String[] args) {
 Scanner sc = new Scanner(System.in);

 System.out.print("Basis: ");
 int a = sc.nextInt();

 System.out.print("Eksponen: ");
 int b = sc.nextInt();

 int hasil = hitungPangkat(a, b);
 System.out.println("Hasil: " + hasil);
 }
}
```

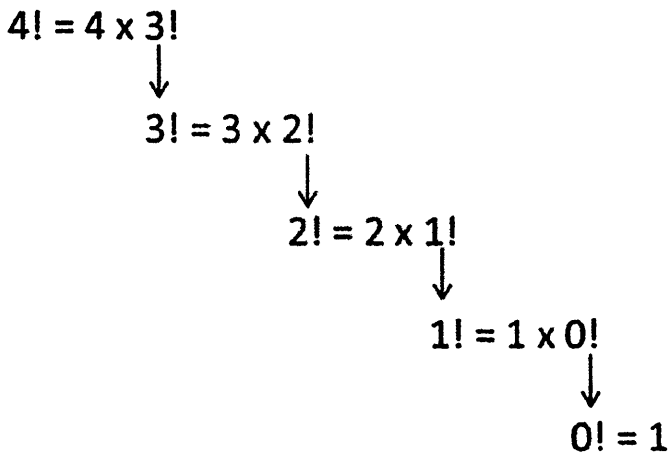
Solusi dalam bahasa Python.

```
def hitungPangkat(basis, eksponen):
 if(eksponen == 0):
 return 1
 else:
 return basis * hitungPangkat(basis, eksponen-1)

def main():
 a = int(input("Basis: "))
 b = int(input("Eksponen: "))
 hasil = hitungPangkat(a, b)
 print("Hasil:", hasil)

if __name__ == '__main__':
 main()
```

Proses yang dijalankan pada kode program di atas dapat dilihat pada ilustrasi berikut.



Contoh kasus 2, buatlah program untuk menghitung hasil faktorial, misal 4! sama dengan 4 x 3 x 2 x 1. Atau dapat juga dituliskan:



Pada kasus ini, proses yang dilakukan berulang adalah proses perkalian dengan kondisi berhenti ketika bilangan yang dihitung faktorialnya adalah bilangan 0. Proses yang akan dijalankan pada Fungsi rekursif adalah perkalian a x (a-1)! dimana nilai a akan selalu dikurangi 1 pada setiap prosesnya, dan proses akan berhenti ketika a=0, dimana Fungsi akan mengembalikan nilai 1.

Solusi dalam bahasa C#.

```
using System;

class Program
{
 static int HitungFaktorial(int nilai)
 {
 if(nilai == 0)
 return 1;
 else
 return nilai * HitungFaktorial(nilai - 1);
```

```
 }

 static void Main(string[] args)
 {
 Console.Write("Bilangan: ");
 int n = int.TryParse(Console.ReadLine());

 double f = HitungFaktorial(n);
 Console.WriteLine("Faktorial: " + f);
 }
}
```

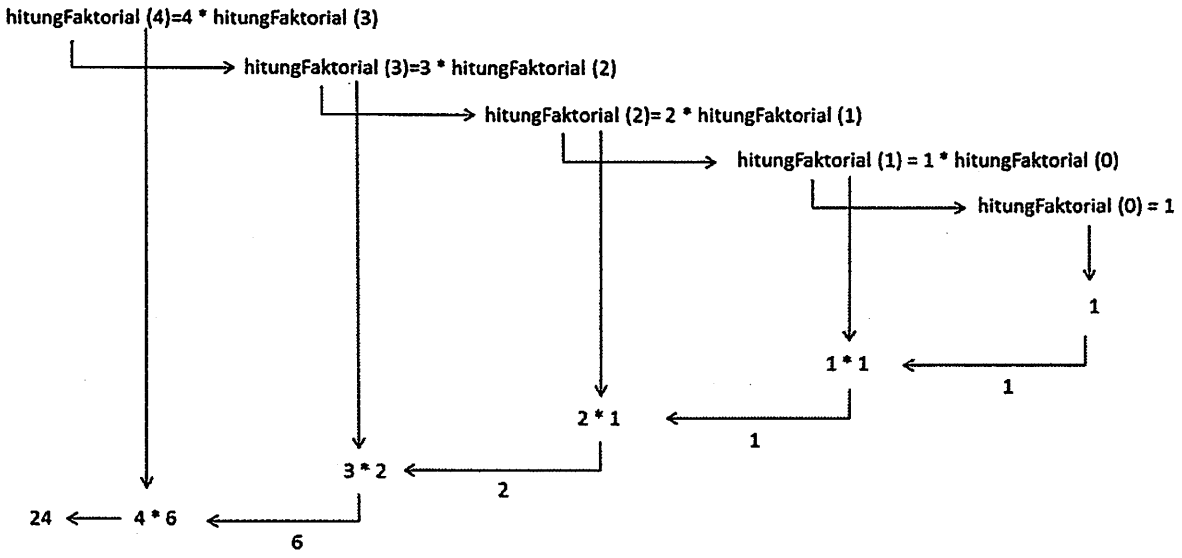
Solusi dalam bahasa JAVA.

```
import java.util.Scanner;

public class Program {
 public static int hitungFaktorial(int nilai) {
 if(nilai == 0)
 return 1;
 else
 return nilai * hitungFaktorial(nilai - 1);
 }

 public static void main(String[] args) {
 Scanner sc = new Scanner(System.in);

 System.out.print("Bilangan: ");
 int n = sc.nextInt();

 int f = hitungFaktorial(n);
 System.out.println("Faktorial: " + f);
 }
}
```

Solusi dalam bahasa Python.

```
def hitungFaktorial(nilai):
 if(nilai == 0):
 return 1
 else:
 return nilai * hitungFaktorial(nilai - 1)

def main():
 n = int(input("Bilangan: "))
 f = hitungFaktorial(n)
 print("Faktorial:", f)

if __name__ == '__main__':
 main()
```

Proses yang dijalankan pada kode program di atas dapat dilihat pada ilustrasi berikut.


hitungFaktorial (4)=4 * hitungFaktorial (3)
hitungFaktorial (3)=3 * hitungFaktorial (2)
hitungFaktorial (2)= 2 * hitungFaktorial (1)
hitungFaktorial (1) = 1 * hitungFaktorial (0)
hitungFaktorial (0) = 1
1
1 * 1
1
2 * 1
1
3 * 2
2
24
4 * 6
6

***

Larik Satu Dimensi

Larik (array) merupakan sebuah variable yang dapat menampung banyak data yang sekelompok dengan tipe data yang sama. Data pada larik tersimpan dalam ruangan-ruangan yang dapat diakses melalui indeks. Indeks sebuah larik merupakan bilangan bulat terurut yang dimulai dari indeks 0 (nol). Kegunaan dari penggunaan larik ini yaitu :

1. Mempermudah dalam penamaan variable untuk data-data yang memiliki kegunaan yang sama.
2. Mempermudah dalam pemesanan memori.
3. Mempermudah dalam pengaksesan data di memori dengan bantuan pengulangan.

Mendefinisikan larik kosong pada bahasa C# dan JAVA, menggunakan sintaks:

```
tipe[] namaVar = new tipe[ukuran];
```

Contoh:

```
int[] nilaiSiswa = new int[10];
```

Sintaks pada bahasa Python:

```
namaVar = [default]*<ukuran>
```

Contoh:

```
nilaiSiswa = [None]*10
```

Nilai default adalah nilai awal yang diberikan pada tiap ruang ketika larik dibuat.

Mendefinisikan larik yang diberi nilai awal pada bahasa C# dan JAVA, menggunakan sintaks:

```
tipe[] namaVar = { nilai1, nilai2, ..., nilaiN };
```

Contoh:

```
int[] nilaiSiswa = { 80, 79, 95, 82);
```

Sintaks pada bahasa Python:

```
namaVar = [ value1, value2, ..., valueN]
```

Contoh:

```
nilaiSiswa = [ 80, 79, 95, 82 ]
```

Untuk dapat mengakses nilai dari setiap ruang pada larik menggunakan perintah namaVar[indeks], hal ini berlaku untuk ketiga bahasa pemrograman. Perintah pengulangan dapat membantu dalam akses nilai dalam larik. Berikut adalah contoh program input/output nilai dari dan ke larik.

Bahasa C#:

```
using System;

class Program
{
 static void Main(string[] args)
 {
 int[] nilai = new int[5];

 for(int i = 0; i < 5; ++i)
 {
 Console.Write("Nilai ke-" + i + ": ");
 nilai[i] = int.Parse(Console.ReadLine());
 }

 Console.WriteLine();
 for(int i = 0; i < 5; ++i)
 Console.Write("Isi larik nilai ke-" + i + ": " + nilai[i]);
 }
}
```

Bahasa JAVA:

```
import java.util.Scanner;

public class Program {
 public static void main(String[] args) {
 Scanner sc = new Scanner(System.in);

 int[] nilai = new int[5];

 for(int i = 0; i < 5; ++i) {
 System.out.print("Nilai ke-" + i + ": ");
 nilai[i] = sc.nextInt();
 }

 Console.WriteLine();
 for(int i = 0; i < 5; ++i)
 System.out.println("Isi larik nilai ke-" + i + ": " + nilai[i]);
 }
}
```

Bahasa Python:

```
def main():
 nilai = [None] * 5
 for i in range(0, 5, 1):
 nilai[i] = int(input("Nilai ke-" + str(i) + ": "))

 for i in range(0, 5, 1):
 print("Isi larik nilai ke-", i, ":", nilai[i])

if __name__ == '__main__':
 main()
```

Proses untuk mengunjungi setiap indeks pada larik (array) disebut traversal. Proses ini dapat membantu dalam menampilkan seluruh data pada lari atau dalam melakukan proses pencarian data. Proses pencarian data akan dijelaskan pada bab mendatang (Algoritma Pencarian).

Contoh lain dari penggunaan larik (array) misalnya untuk mencari nilai terbesar dan

terkecil dari sekelompok bilangan. Data bilangan yang akan dibandingkan disimpan dalam larik (array), kemudian kita akan melakukan perbandingan, mulai dari indeks pertama sampai indeks paling terakhir.

Solusi pada bahasa C#:

```
using System;

class Program
{
 static void Main(string[] args)
 {
 int[] bilangan = new int[6];
 for(int i = 0; i < 6; ++i)
 {
 Console.Write("Bilangan ke-" + i + ": ");
 bilangan[i] = int.Parse(Console.ReadLine());
 }

 int nilaiMin = bilangan[0];
 int nilaiMax = bilangan[0];
 for(int i = 1; i < 6; ++i)
 {
 if(bilangan[i] < nilaiMin)
 nilaiMin = bilangan[i];
 if(bilangan[i] > nilaiMax)
 nilaiMax = bilangan[i]
 }

 Console.WriteLine("Nilai terkecil: " + nilaiMin);
 Console.WriteLine("Nilai terbesar: " + nilaiMax);
 }
}
```

Solusi pada bahasa JAVA:

```
import java.util.Scanner;

public class Program {
 public static void main(String[] args) {
 Scanner sc = new Scanner(System.in);

 int[] bilangan = new int[6];
 for(int i = 0; i < 6; ++i) {
 System.out.print("Bilangan ke-" + i + ": ");
 bilangan[i] = sc.nextInt();
 }

 int nilaiMin = bilangan[0];
 int nilaiMax = bilangan[0];
 for(int i = 1; i < 6; ++i) {
 if(bilangan[i] < nilaiMin)
 nilaiMin = bilangan[i];
 if(bilangan[i] > nilaiMax)
 nilaiMax = bilangan[i]
 }

 System.out.println("Nilai terkecil: " + nilaiMin);
 System.out.println("Nilai terbesar: " + nilaiMax);
 }
}
```

Solusi pada bahasa Python:

```
def main():
 bilangan = [None] * 6
 for i in range(0, 6, 1):
 bilangan[i] = int(input("Bilangan ke-" + str(i) + ": "))

 nilaiMin = bilangan[0]
 nilaiMax = bilangan[0]
 for i in range(1, 6, 1):
 if(bilangan[i] < nilaiMin):
 nilaiMin = bilangan[i]
 if(bilangan[i] > nilaiMax):
 nilaiMax = bilangan[i]

 print("Nilai terkecil:", nilaiMin)
 print("Nilai terbesar:", nilaiMax)

if __name__ == '__main__':
 main()
```

Proses pencarian bilangan terbesar dan terkecil dapat dilihat pada tabel berikut.

Counter (i)	Pembanding	nilaiMin	nilaiMax
		10	10
1	bilangan[1]=14	14<10 (False)	14>10 (True) -> nilaiMax=14
2	bilangan[2]=8	8<10 (True) -> nilaiMin=8	8>14 (False)
3	bilangan[3]=20	20<8 (False)	20>14 (True) -> nilaiMax=20
4	bilangan[4]=1	1<8 (True) -> nilaiMin=1	1>20 (False)
5	bilangan[5]=9	9<1 (False)	9>20 (False)

***

Array of Character (String)

String dalam bahasa pemrograman merupakan susunan dari karakter (char), dimana setiap karakter yang ada dapat diakses dengan menggunakan indeks.

Contoh, String kata="Welcome", ilustrasi dalam bentuk array of character :

W	e	l	c	o	m	e
0	1	2	3	4	5	6

kata[0] = "W"
kata[1] = "e"
kata[2] = "l"
kata[3] = "c"
kata[4] = "o"
kata[5] = "m"
kata[6] = "e"

Contoh kasus, buatlah program untuk menghitung banyaknya huruf vokal dala sebuah kalimat yang dimasukan oleh pengguna. Berikut solusi dalam bahasa C#.

```
using System;

class Program
{
 static void Main(string[] args)
 {
 Console.Write("Masukan sebuah kalimat diakhiri tanda titik: ");
 string kalimat = Console.ReadLine();

 int vokal = 0, i = 0;
 while(kalimat[i] != '.') {
 if(kalimat[i] == 'a' || kalimat[i] == 'i' || kalimat[i] == 'u'
 || kalimat[i] == 'e' || kalimat[i] == 'o')
 vokal++;

 i++;
 }

 Console.WriteLine("Jumlah huruf vokal: " + vokal);
 }
}
```

Solusi dalam bahasa JAVA.

```
import java.util.Scanner;

public class Program {
 public static void main(String[] args) {
 Scanner sc = new Scanner(System.in);

 System.out.print("Masukan sebuah kalimat diakhiri tanda titik: ");
 String kalimat = sc.next();

 int vokal = 0, i = 0;
 while(kalimat[i] != '.') {
 if(kalimat[i] == 'a' || kalimat[i] == 'i' || kalimat[i] == 'u'
 || kalimat[i] == 'e' || kalimat[i] == 'o')
```

```
 vokal++;

 i++;
 }

 System.out.println("Jumlah huruf vokal: " + vokal);
 }
}
```

Solusi dalam bahasa Python.

```
def main():
 kalimat = input("Masukan sebuah kalimat diakhiri tanda titik:")
 vokal = 0
 i = 0
 while(kalimat[i] != '.'):
 if(kalimat[i] == 'a' or kalimat[i] == 'i' or kalimat[i] == 'u'
 or kalimat[i] == 'e' or kalimat[i] == 'o'):
 vokal += 1
 i += 1

 print("Jumlah huruf vokal:", vokal)

if __name__ == '__main__':
 main()
```

***

Array dan Fungsi

Penggunaan larik (array) pada Fungsi sama halnya ketika menggunakan pada fungsi main(). Larik (array) pun dapat dijadikan sebagai parameter atau nilai kembalian (return) dari sebuah Fungsi. Berikut adalah contoh program yang bertujuan untuk inisialisasi dan menjumlahkan nilai dalam larik menggunakan fungsi.

Bahasa C#:

```
using System;

class Program
{
 static int[] IsiArray(int n)
 {
 int[] arr = new int[n];
 for(int i = 0; i < n; ++i)
 arr[i] = int.Parse(Console.ReadLine());
 return arr;
 }

 int HitungSum(int[] arr)
 {
 int sum = 0;
 for(int i = 0; i < arr.Length; ++i)
 sum += arr[i];
 return sum;
 }


 static void Main(string[] args)
 {
 Console.Write("Jumlah bilangan: ");
 int n = int.Parse(Console.ReadLine());

 int[] bil = IsiArray(n);
 Console.WriteLine("Hasil sum: " + HitungSum(bil));
 }
}
```

Bahasa JAVA:

```
import java.util.Scanner;

public class Program {
 public static int[] isiArray(int n) {
 Scanner sc = new Scanner(System.in);
 int[] arr = new int[n];
 for(int i = 0; i < n; ++i)
 arr[i] = sc.nextInt();
 return arr;
 }

 public static int hitungSum(int[] arr) {
 int sum = 0;
 for(int i = 0; i < arr.length; ++i)
 sum += arr[i];
 return sum;
 }

 public static void main(String[] args) {
```

```
 Scanner sc = new Scanner(System.in);

 System.out.print("Jumlah bilangan: ");
 int n = sc.nextInt();

 int[] bil = isiArray(n);
 System.out.println("Hasil sum: " + hitungSum(bil));
 }
}
```

Bahasa Python:

```
def isiArray(n):
 arr = [None] * n
 for i in range(0, n, 1):
 arr[i] = int(input())
 return arr

def hitungSum(arr, n):
 sum = 0
 for i in range(0, n, 1):
 sum += arr[i]
 return sum

def main():
 n = int(input("Jumlah bilangan:"))
 bil = [None] * n
 bil = isiArray(n)
 print("Hasil sum: ", hitungSum(bil, n))

if __name__ == '__main__':
 main()
```

***

Larik Dua Dimensi

Larik (array) dapat juga dibuat multi dimensi. Pada umumnya larik yang digunakan ialah larik satu dimensi dan larik dua dimensi (matriks). Sama seperti larik satu dimensi, nilai dari larik dua dimensi diakses melalui indeks. Perbedaannya ialah pada larik dua dimensi terdapat dua buah indeks, yaitu baris dan kolom.

Berikut adalah ilustrasi dari larik dua dimensi. Larik ini memiliki 3 baris dan 4 kolom. Perhatikan pemberian indeks pada masing-masing ruangan.

	0	1	2	3
0	[0][0]	[0][1]	[0][2]	[0][3]
1	[1][0]	[1][1]	[1][2]	[1][3]
2	[2][0]	[2][1]	[2][2]	[2][3]

Sama halnya dengan larik satu dimensi, larik harus didefinisikan terlebih dahulu berapa banyak ruang yang dibutuhkan. Berikut ini cara untuk mendefinisikan larik dua dimensi, untuk tiap bahasa pemrograman.

Mendefinisikan larik kosong dalam bahasa C#.

```
tipe[,] namaVar = new tipe[jumlahBaris, jumlahKolom];
```

Contoh:

```
int[,] matriks = new int[3, 4];
```

Bahasa JAVA.

```
tipe[][] namaVar = new tipe[jumlahBaris][jumlahKolom];
```

Contoh:

```
int[][] matriks = new int[3][4];
```

Bahasa Python.

```
namaVar = [[default] * jumlahKolom for i in range(jumlahBaris)]
```

Contoh:

```
matriks = [[None] * 4 for i in range(3)]
```

Berikut adalah cara mendefinisikan larik yang telah diberi nilai awal dalam bahasa C#.

```
tipe[,] namaVar = {{nilai1, nilai2, ...}, {...}, ...};
```

Contoh:

```
int[,] matriks = {{10,12,9,14},{12,8,3,11},{7,32,16,2}};
```

Bahasa JAVA.

```
tipe[][] namaVar = {{nilai1, nilai2, ...}, {...}, ...};
```

Contoh:

```
int[][] matriks = {{10,12,9,14},{12,8,3,11},{7,32,16,2}};
```

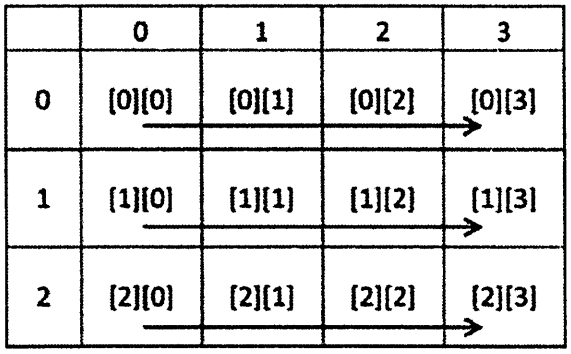
Bahasa Python.

```
namaVar = [[nilai1, nilai2, ...], [...], ...]
```

Contoh

```
matriks = [[10,12,9,14],[12,8,3,11],[7,32,16,2]]
```

Berbeda sedikit dengan larik satu dimensi, untuk mengakses larik dua dimensi dapat menggunakan bantuan pengulangan bersarang. Berikut adalah ilustrasi pengisian matriks yang berjalan per baris dari kolom indeks pertama sampai terakhir.



Solusi bahasa C#.

```
using System;

class Program
{
 static void Main(string[] args)
 {
 int[,] matriks = new int[3, 3];
 for(int b = 0; b < 3, ++b)
 {
 for(int k = 0; k < 3; ++k)
 {
 Console.Write("Array[" + b + "][" + k + "]: ");
 matriks[b,k] = int.Parse(Console.ReadLine());
 }
 }

 for(int b = 0; b < 3, ++b)
 {
 for(int k = 0; k < 3; ++k)
 Console.Write(matriks[b,k] + " ");
 Console.WriteLine();
 }
 }
}
```

Solusi bahasa JAVA.

```
import java.util.Scanner;

public class Program {
 public static void main(String[] args) {
 Scanner sc = new Scanner(System.in);

 int[][] matriks = new int[3][3];
 for(int b = 0; b < 3, ++b) {
 for(int k = 0; k < 3; ++k) {
 System.out.print("Array[" + b + "][" + k + "]: ");
 matriks[b][k] = sc.nextInt();
 }
 }

 for(int b = 0; b < 3, ++b) {
 for(int k = 0; k < 3; ++k) {
 System.out.print(matriks[b][k] + " ");
 }
 System.out.println();
 }
 }
}
```

Solusi bahasa Python.

```
def main():
 matriks = [[0] * 3 for i in range(3)]

 for b in range(0, 3, 1):
 for k in range(0, 3, 1):
 matriks[b][k] = int(input("Array[" + str(b) + "][" + str(k) + "]:"))

 print("Isi matriks:")
 for b in range(0, 3, 1):
 for k in range(0, 3, 1):
 print(matriks[b][k], end = " ")
 print()

if __name__ == '__main__':
 main()
```

Contoh kasus, buatlah sebuah program untuk menjumlahkan dua buah matriks dengan ordo 3x3. Dua matriks telah ditentukan nilainya pada ilustrasi berikut.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} + \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 10 & 10 & 10 \\ 10 & 10 & 10 \\ 10 & 10 & 10 \end{bmatrix}$$

Solusi bahasa C#.

```
using System;

class Program
{
 static void Main(string[] args)
```

```
 {
 int[,] matA = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
 int[,] matB = {{9, 8, 7}, {6, 5, 4}, {3, 2, 1}};
 int[,] matC = new int[3, 3];
 for(int b = 0; b < 3, ++b)
 for(int k = 0; k < 3; ++k)
 matC[b,k] = matA[b, k] + matB[b, k];

 Console.WriteLine("Hasil penjumlahan:");
 for(int b = 0; b < 3, ++b)
 {
 for(int k = 0; k < 3; ++k)
 Console.Write(matC[b,k] + " ");
 Console.WriteLine();
 }
 }
}
```

Solusi bahasa JAVA.

```
import java.util.Scanner;

public class Program {
 public static void main(String[] args) {
 Scanner sc = new Scanner(System.in);

 int[][] matA = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
 int[][] matB = {{9, 8, 7}, {6, 5, 4}, {3, 2, 1}};
 int[][] matC = new int[3][3];
 for(int b = 0; b < 3, ++b)
 for(int k = 0; k < 3; ++k)
 matC[b][k] = matA[b][k] + matB[b][k];

 System.out.println("Hasil penjumlahan:");
 for(int b = 0; b < 3, ++b) {
 for(int k = 0; k < 3; ++k) {
 System.out.print(matC[b][k] + " ");
 }
 System.out.println();
 }
 }
}
```

Solusi bahasa Python.

```
def main():
 matA = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
 matB = [[9, 8, 7], [6, 5, 4], [3, 2, 1]]
 matC = [[0] * 3 for i in range(3)]

 for b in range(0, 3, 1):
 for k in range(0, 3, 1):
 matC[b][k] = matA[b][k] + matB[b][k]

 print("Hasil penjumlahan:")
 for b in range(0, 3, 1):
 for k in range(0, 3, 1):
 print(matC[b][k], end = " ")
 print()

if __name__ == '__main__':
 main()
```

Contoh kasus, buatlah sebuah program yang menjumlahkan isi array pada setiap baris dan kolom.

	0	1	2	
0	1	2	3	= 6
1	4	5	6	= 15
2	7	8	9	= 24
	= 12	= 15	= 18	

Solusi bahasa C#.

```
using System;

class Program
{
 static void Main(string[] args)
 {
 int[,] mat = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
 for(int b = 0; b < 3, ++b)
 {
 int sum = 0;
 for(int k = 0; k < 3; ++k)
 sum += mat[b, k];
 Console.WriteLine("Jumlah pada baris " + b + ": " + sum);
 }

 for(int k = 0; k < 3, ++b)
 {
 int sum = 0;
 for(int b = 0; b < 3; ++k)
 sum += mat[b, k];
 Console.WriteLine("Jumlah pada kolom " + k + ": " + sum);
 }
 }
}
```

Solusi bahasa JAVA.

```
import java.util.Scanner;

public class Program {
 public static void main(String[] args) {
 Scanner sc = new Scanner(System.in);

 int[][] mat = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
 for(int b = 0; b < 3, ++b) {
 int sum = 0;
 for(int k = 0; k < 3; ++k)
 sum += mat[b][k];
 System.out.println("Jumlah pada baris " + b + ": " + sum);
```

```
 }

 for(int k = 0; k < 3, ++k) {
 int sum = 0;
 for(int b = 0; b < 3; ++b) {
 sum += mat[b][k];
 System.out.println("Jumlah pada kolom " + k + ": " + sum);
 }
 }
}
```

Solusi bahasa Python.

```
def main():
 mat = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

 for b in range(0, 3, 1):
 sum = 0;
 for k in range(0, 3, 1):
 sum += mat[b][k]
 print("Jumlah pada baris", b, ":", sum);

 for k in range(0, 3, 1):
 for b in range(0, 3, 1):
 sum += mat[b][k]
 print("Jumlah pada kolom", k, ":", sum);

if __name__ == '__main__':
 main()
```

***

Algoritma Pencarian

Ketika terdapat banyak data yang telah terkumpul untuk menemukan sebuah data diperlukan metode pencarian. Contohnya pencarian nomor telepon pada buku kontak, pencarian jadwal pesawat, pencarian data siswa, dan lain sebagainya. Algoritma pencarian yang akan dibahas pada bab ini ialah Sequential Search dan Binary Search.

***

Sequential Search

Sequential Search merupakan metode pencarian dengan melakukan pengecekan data satu per satu pada larik (array) mulai dari indeks ke nol. Terdapat tiga macam pencarian dengan metode ini, yaitu pencarian pada tabel tidak terurut, pencarian pada tabel terurut, dan pencarian dengan sentinel.

1. Pencarian pada tabel tidak terurut.
Metode ini digunakan ketika data pada larik (array) tidak terurut. Pada metode ini, kita akan mengunjungi setiap indeks yang ada pada larik (array). Kunjungan dimulai dari mengunjungi indeks nol. Hal ini dimaksudkan untuk melakukan pengecekan, apakah data pada indeks tersebut sama dengan data yang sedang dicari atau tidak. Jika tidak, maka pencarian akan dilanjutkan pada indeks berikutnya. Pencarian akan berlangsung sampai data ditemukan atau ketika sudah mencapai indeks paling terakhir.

24	11	15	7	32	18	21	27
0	1	2	3	4	5	6	7

Contoh 1, bilCari=32, pengecekan dilakukan terhadap {24,11,15,7,32} dan karena ditemukan maka, pencarian berhenti pada indeks 4.

indeks	Perbandingan
0	24 = 32? (False)
1	11 = 32? (False)
2	15 = 32? (False)
3	7 = 32? (False)
4	32 = 32? (True) -> STOP (data ditemukan)

Contoh 2, bilCari=50, pengecekan dilakukan terhadap {24,11,15,7,32,18,21,27}. Pencarian baru berhenti setelah program melakukan pengecekan untuk data pada indeks 7.

indeks	Perbandingan
0	24 = 50? (False)
1	11 = 50? (False)
2	15 = 50? (False)
3	7 = 50? (False)
4	32 = 50? (False)
5	18 = 50? (False)
6	21 = 50? (False)
7	27 = 50? (False) -> STOP (indeks paling akhir)

Solusi bahasa C#.

```
using System;

class Program
```

```
{
 static bool SeqSearch(int[] data, int cari)
 {
 bool found = false;
 int i = 0;
 while(!found && i < data.Length)
 {
 if(data[i] == cari)
 found = true;
 else
 i += 1;
 }
 return found;
 }

 static void Main(string[] args)
 {
 int[] bilangan = {24, 11, 15, 7, 32, 18, 21, 27};

 Console.Write("Bilangan yang dicari: ");
 int bilCari = int.Parse(Console.ReadLine());

 bool ditemukan = SeqSearch(bilangan, bilCari);
 if(ditemukan)
 Console.WriteLine("Data ditemukan.");
 else
 Console.WriteLine("Data tidak ditemukan."); }
}
```

Solusi bahasa JAVA.

```
import java.util.Scanner;

public class Program {
 public static boolean seqSearch(int[] data, int cari) {
 boolean found = false;
 int i = 0;
 while(!found && i < data.length) {
 if(data[i] == cari)
 found = true;
 else
 i += 1;
 }
 return found;
 }

 public static void main(String[] args) {
 Scanner sc = new Scanner(System.in);

 int[] bilangan = {24, 11, 15, 7, 32, 18, 21, 27};

 System.out.print("Bilangan yang dicari: ");
 int bilCari = sc.nextInt();

 boolean ditemukan = seqSearch(bilangan, bilCari);
 if(ditemukan)
 System.out.println("Data ditemukan.");
 else
 System.out.println("Data tidak ditemukan.");
 }
}
```

Solusi bahasa Python.

```
def seqSearch(n, data, cari):
 found = False
 i = 0
 while(not found and i < n):
 if(data[i] == cari):
 found = True
 else:
 i += 1
 return found

def main():
 n = 8
 bilangan = [24, 11, 15, 7, 32, 18, 21, 27]
 bilCari = int(input("Bilangan yang dicari:"))

 ditemukan = seqSearch(n, bilangan, bilCari)
 if(ditemukan):
 print("Data ditemukan.")
 else:
 print("Data tidak ditemukan.")

if __name__ == '__main__':
 main()
```

2. Pencarian pada tabel terurut.

Metode ini digunakan ketika data pada larik (array) sudah terurut. Pada metode ini, kita akan mengunjungi setiap indeks yang ada pada larik (array). Kunjungan dimulai dari mengunjungi indeks nol. Berbeda sedikit dengan metode sebelumnya, pada metode ini pencarian akan dilakukan selama data yang ditemukan lebih kecil/lebih besar dari data yang dicari. Jika data terurut dari kecil ke besar, maka pencarian akan dilakukan selama data yang ditemukan lebih kecil dari data yang dicari atau indeks sudah berada pada indeks terakhir. Sedangkan jika data terurut dari besar ke kecil, maka pencarian akan dilakukan selama data yang ditemukan lebih besar dari data yang dicari atau indeks sudah berada pada indeks terakhir. Setelah pencarian berhenti, lakukan pengecekan data pada indeks dimana pencarian berhenti.

32	27	24	21	18	15	11	7
0	1	2	3	4	5	6	7

Contoh 1, bilCari=21, pengecekan dilakukan terhadap {32,27,24,21} dan karena ditemukan maka, pencarian berhenti pada indeks 3.

indeks	Perbandingan
0	32 > 21? (False)
1	27 > 21? (False)
2	24 > 21? (False)
3	21 > 21? (False) -> STOP (data > 21)

Pencarian berhenti pada indeks 3. Langkah berikutnya adalah melakukan pengecekan, apakah data pada indeks 3 sama dengan data yang dicari?

Indeks ke 3 : 21 = 21? (True) Data ditemukan.

Contoh 2, bilCari=20, pengecekan dilakukan terhadap {32,27,24,21,18,15,11,7}.

Pencarian baru berhenti setelah program melakukan pengecekan untuk data pada indeks 7.

indeks	Perbandingan
0	32 > 20? (False)
1	27 > 20? (False)
2	24 > 20? (False)
3	21 > 20? (False)
4	18 > 20? (False) -> STOP (data > 21)

Pencarian berhenti pada indeks 4. Langkah berikutnya adalah melakukan pengecekan, apakah data pada indeks 4 sama dengan data yang dicari?

Indeks ke 4 : 18 = 20? (False) Data tidak ditemukan.

Solusi bahasa C#.

```
using System;

class Program
{
 static bool SeqSearchSorted(int[] data, int cari)
 {
 int i = 0;
 while(data[i] > cari && i < data.Length-1)
 i++;
 if(data[i] == cari)
 return true;
 else
 return false;
 }

 static void Main(string[] args)
 {
 int[] bilangan = {32, 27, 24, 21, 18, 15, 11, 7};

 Console.Write("Bilangan yang dicari: ");
 int bilCari = int.Parse(Console.ReadLine());

 bool ditemukan = SeqSearchSorted(bilangan, bilCari);
 if(ditemukan)
 Console.WriteLine("Data ditemukan.");
 else
 Console.WriteLine("Data tidak ditemukan."); }
}
```

Solusi bahasa JAVA.

```
import java.util.Scanner;

public class Program {
 public static boolean seqSearchSorted(int[] data, int cari) {
 int i = 0;
 while(data[i] > cari && i < data.length-1)
 i++;
 if(data[i] == cari)
 return true
```

```
 else
 return false;
 }

 public static void main(String[] args) {
 Scanner sc = new Scanner(System.in);

 int[] bilangan = {32, 27, 24, 21, 18, 15, 11, 7};

 System.out.print("Bilangan yang dicari: ");
 int bilCari = sc.nextInt();

 boolean ditemukan = seqSearchSorted(bilangan, bilCari);
 if(ditemukan)
 System.out.println("Data ditemukan.");
 else
 System.out.println("Data tidak ditemukan.");
 }
}
```

Solusi bahasa Python.

```
def seqSearchSorted(n, data, cari):
 i = 0
 while(data[i] > cari and i < n-1):
 i += 1
 if(data[i] == cari):
 return True
 else:
 return False

def main():
 n = 8
 bilangan = [32, 27, 24, 21, 18, 15, 11, 7]
 bilCari = int(input("Bilangan yang dicari:"))

 ditemukan = seqSearchSorted(n, bilangan, bilCari)
 if(ditemukan):
 print("Data ditemukan.")
 else:
 print("Data tidak ditemukan.")

if __name__ == '__main__':
 main()
```

3. Pencarian dengan sentinel.

Metode ini menggunakan tambahan ruang pada larik (array) pada indeks paling terakhir, ruang tambahan inilah yang disebut dengan sentinel. Ruang sentinel ini akan berisi nilai dari data yang sedang dicari. Pencarian data akan dilakukan dari mulai indeks ke nol hingga data ditemukan. Ketika data dinyatakan ditemukan, maka langkah berikutnya adalah melakukan pengecekan terhadap indeks data ditemukan. Jika indeks data ditemukan pada indeks data sentinel, maka berarti data tidak ada pada data asli. Sedangkan jika data ditemukan pada indeks yang lebih kecil dari indeks sentinel, artinya data itu memang ada pada data larik (array).

24	11	15	7	32	18	21	27	**(sentinel)**
0	1	2	3	4	5	6	7	**8**

Contoh 1, bilCari=7.

24	11	15	7	32	18	21	27	**7**
0	1	2	3	4	5	6	7	**8**

indeks	Perbandingan
0	24 = 7? (False)
1	11 = 7? (False)
2	15 = 7? (False)
3	7 = 7? (True) -> STOP (data ditemukan)

Langkah berikutnya, lakukan pengecekan pada indeks data ditemukan. Pencarian berhenti pada indeks 3, sedangkan sentinel berada pada indeks 8. Ini berarti data ditemukan pada data asli.

Contoh 2, bilCari=90.

24	11	15	7	32	18	21	27	**90**
0	1	2	3	4	5	6	7	**8**

indeks	Perbandingan
0	24 = 90? (False)
1	11 = 90? (False)
2	15 = 90? (False)
3	7 = 90? (False)
4	32 = 90? (False)
5	18 = 90? (False)
6	21 = 90? (False)
7	27 = 90? (False)
8	90 = 90? (True) -> STOP (data ditemukan)

Langkah berikutnya, lakukan pengecekan pada indeks data ditemukan. Pencarian berhenti pada indeks 8, sedangkan sentinel berada pada indeks 8. Ini berarti data tidak ditemukan pada data asli.

Solusi bahasa C#.

```
using System;

class Program
{
 static bool SeqSearchSentinel(int[] data, int cari)
 {
 data[data.Length-1] = cari;
 int i = 0;
 while(data[i] != cari)
 i++;
 if(i < data.Length-1)
 return true;
 else
 return false;
 }
```

```
 static void Main(string[] args)
 {
 int[] bilangan = {32, 27, 24, 21, 18, 15, 11, 7, 0};

 Console.Write("Bilangan yang dicari: ");
 int bilCari = int.Parse(Console.ReadLine());

 bool ditemukan = SeqSearchSentinel(bilangan, bilCari);
 if(ditemukan)
 Console.WriteLine("Data ditemukan.");
 else
 Console.WriteLine("Data tidak ditemukan.");
 }
}
```

Solusi bahasa JAVA.

```
import java.util.Scanner;

public class Program {
 public static boolean seqSearchSentinel(int[] data, int cari) {
 data[data.length-1] = cari;
 int i = 0;
 while(data[i] != cari)
 i++;
 if(i < data.length-1)
 return true
 else
 return false;
 }

 public static void main(String[] args) {
 Scanner sc = new Scanner(System.in);

 int[] bilangan = {32, 27, 24, 21, 18, 15, 11, 7, 0};

 System.out.print("Bilangan yang dicari: ");
 int bilCari = sc.nextInt();

 boolean ditemukan = seqSearchSentinel(bilangan, bilCari);
 if(ditemukan)
 System.out.println("Data ditemukan.");
 else
 System.out.println("Data tidak ditemukan.");
 }
}
```

Solusi bahasa Python.

```
def seqSearchSentinel(n, data, cari):
 data[n] = cari
 i = 0
 while(data[i] != cari):
 i += 1
 if(i < n):
 return True
 else:
 return False

def main():
 n = 8
```

```
    bilangan = [32, 27, 24, 21, 18, 15, 11, 7, None]
    bilCari = int(input("Bilangan yang dicari:"))

    ditemukan = seqSearchSentinel(n, bilangan, bilCari)
    if(ditemukan):
    print("Data ditemukan.")
    else:
    print("Data tidak ditemukan.")

if __name__ == '__main__':
    main()
```

***

Binary Search

Metode Binary Search merupakan metode pencarian dengan membagi dua ruang pencarian secara berkelanjutan sehingga dapat memperkecil ruang pencarian dan mengurangi jumlah operasi perbandingan. Metode Binary Search ini digunakan pada data yang sudah terurut baik dari kecil ke besar, maupun terurut dari besar ke kecil.

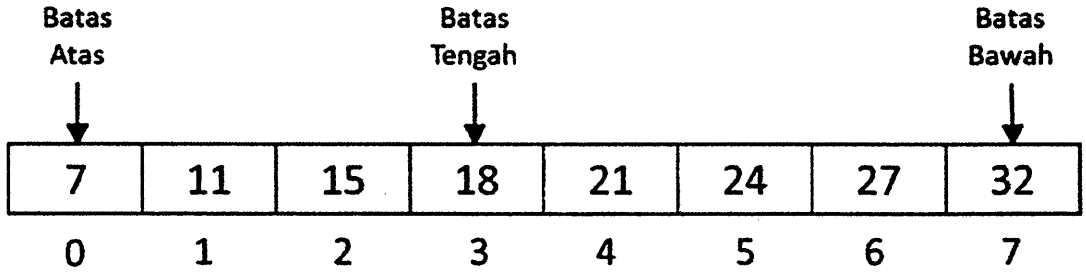
Pada contoh dibawah ini, kita akan menggunakan data yang sudah terurut dari kecil ke besar. Proses yang akan dilakukan oleh metode pencarian ini adalah sebagai berikut:

1. Tentukanlah indeks yang akan dijadikan sebagai BatasAtas dan BatasBawah. Pada proses awal, BatasAtas = 0 dan BatasBawah = N-1.
2. Tentukan BatasTengah, yaitu (BatasAtas + BatasBawah)/2.
3. Lakukan pengecekan pada element di indeks BatasTengah, apakah data sama dengan data yang dicari atau tidak. Jika sama, maka data ditemukan dan pencarian dihentikan. Tetapi, jika data tidak sama, maka:
 a. Jika element BatasTengah lebih kecil dari data yang dicari maka, BatasAtas = BatasTengah + 1.
 b. Jika element BatasTengah lebih besar dari data yang dicari maka, BatasBawah = BatasTengah – 1.
4. Lakukan kembali langkah 2- 4 selama Batas Atas <= Batas Bawah dan data belum ditemukan.

7	11	15	18	21	24	27	32
0	1	2	3	4	5	6	7

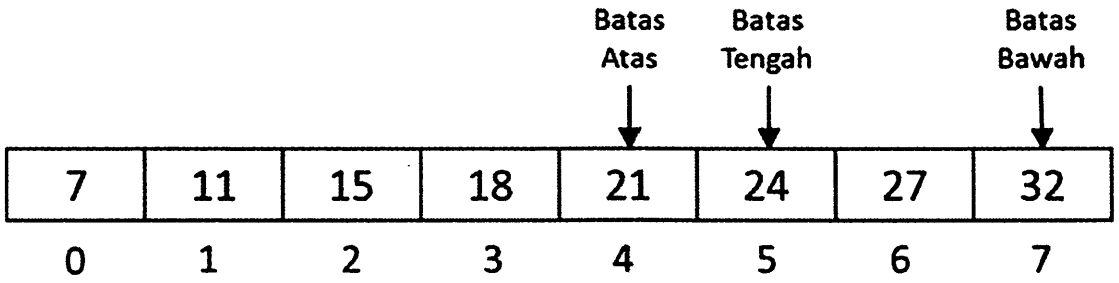
Contoh 1, lakukan pencarian untuk bilangan 27, nilaiCari = 27.

Langkah 1:



BatasAtas	BatasBawah	BatasTengah	Pengecekan	Hasil
0	7	(0+7)/2 = 3	arr[3] = 18 18 = 27? (False) 18...27? <	BatasTengah < nilaiCari, maka: BatasAtas = BatasTengah + 1 BatasBawah = 3 + 1 = 4

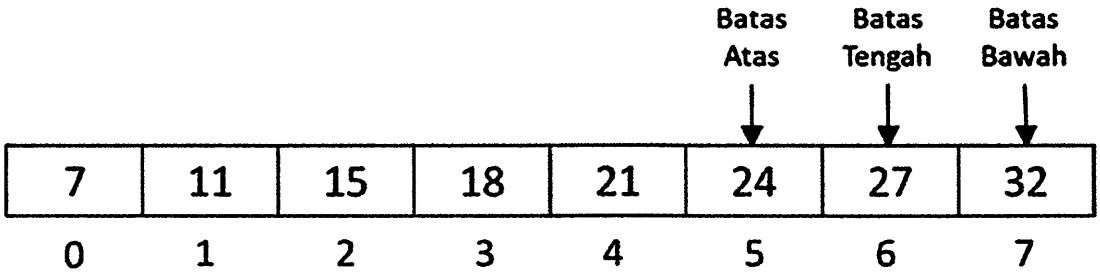
Langkah 2:



BatasAtas <= BatasBawah? (True) Proses lanjut.

BatasAtas	BatasBawah	BatasTengah	Pengecekan	Hasil
4	7	(4+7)/2 = 5	arr[5] = 24 24 = 27? (False) 24...27? <	BatasTengah < nilaiCari, maka: BatasAtas = BatasTengah + 1 BatasBawah = 4 + 1 = 5

Langkah 3:

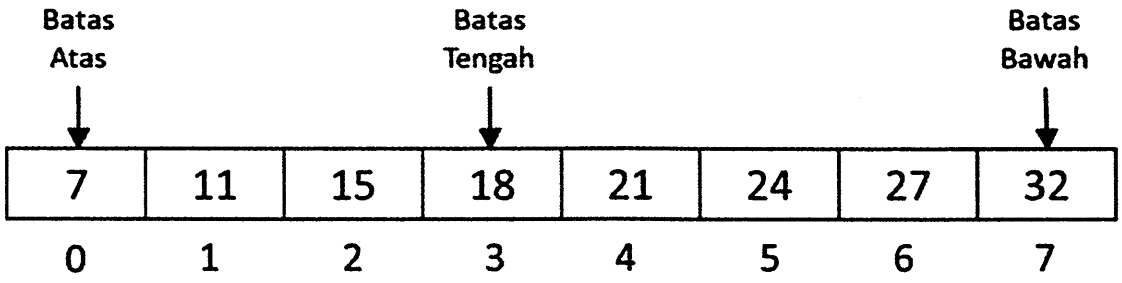


BatasAtas <= BatasBawah? (True) Proses lanjut.

BatasAtas	BatasBawah	BatasTengah	Pengecekan	Hasil
5	7	(5+7)/2 = 6	arr[6] = 27 27 = 27? (True)	BatasTengah = nilaiCari, maka: Data ditemukan, proses berhenti

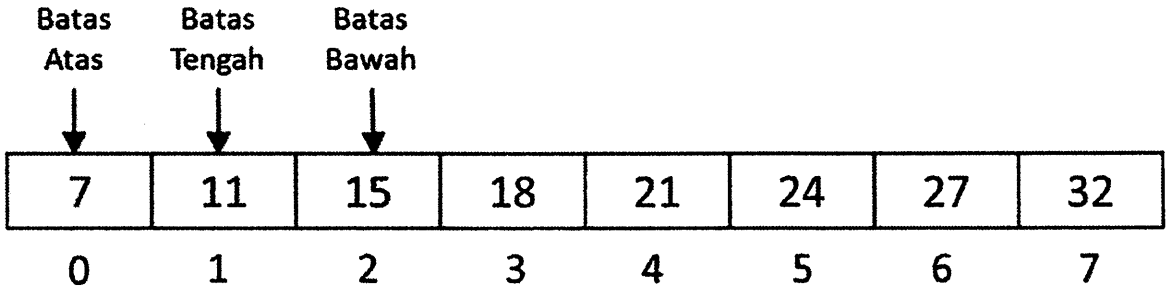
Contoh 2, lakukan pencarian untuk bilangan 12, nilaiCari = 12.

Langkah 1:



BatasAtas	BatasBawah	BatasTengah	Pengecekan	Hasil
0	7	(0+7)/2 = 3	arr[3] = 18 18 = 12? (False) 18...12? >	BatasTengah > nilaiCari, maka: BatasBawah = BatasTengah - 1 BatasAtas = 3 - 1 = 2

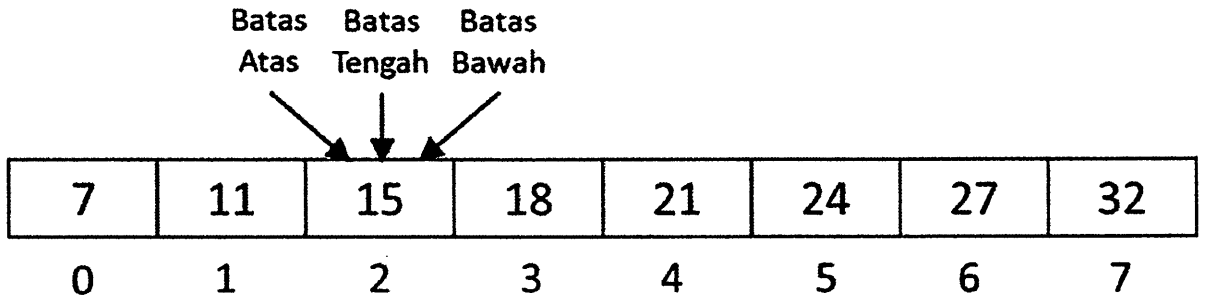
Langkah 2:



BatasAtas <= BatasBawah? (True) Proses lanjut.

BatasAtas	BatasBawah	BatasTengah	Pengecekan	Hasil
0	2	(0+2)/2 = 1	arr[1] = 11 11 = 12? (False) 11...12? <	BatasTengah < nilaiCari, maka: BatasAtas = BatasTengah + 1 BatasBawah = 1 + 1 = 2

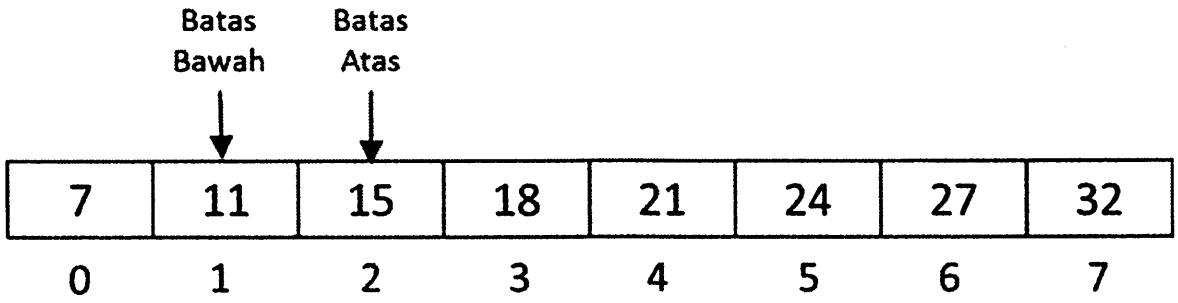
Langkah 3:



BatasAtas <= BatasBawah? (True) Proses lanjut.

BatasAtas	BatasBawah	BatasTengah	Pengecekan	Hasil
2	2	(2+2)/2 = 1	arr[2] = 15 15 = 12? (False) 15...12? >	BatasTengah > nilaiCari, maka: BatasBawah = BatasTengah - 1 BatasAtas = 2 - 1 = 1

Langkah 4:



BatasAtas <= BatasBawah? (False) Proses dihentikan, data tidak ditemukan.

Solusi bahasa C#.

```
using System;

class Program
{
 static bool BinarySearch(int[] data, int cari)
 {
 bool found = false;
 int atas = 0, bawah = data.length-1;
 while(atas <= bawah && !found)
 {
 tengah = (atas + bawah) / 2;
 if(data[tengah] == cari)
 found = true;
 else if(cari < data[tengah])
 bawah = tengah - 1;
 else
 atas = tengah + 1;
 }
 return found;
 }

 static void Main(string[] args)
 {
 int[] bilangan = {7, 11, 15, 18, 21, 24, 27, 32};

 Console.Write("Bilangan yang dicari: ");
 int bilCari = int.Parse(Console.ReadLine());

 bool ditemukan = BinarySearch(bilangan, bilCari);
 if(ditemukan)
 Console.WriteLine("Data ditemukan.");
 else
 Console.WriteLine("Data tidak ditemukan."); }
}
```

Solusi bahasa JAVA.

```
import java.util.Scanner;

public class Program {
 public static boolean binarySearch(int[] data, int cari) {
 boolean found = false;
 int atas = 0, bawah = data.length-1;
 while(atas <= bawah && !found) {
 tengah = (atas + bawah) / 2;
```

```
 if(data[tengah] == cari)
 found = true;
 else if(cari < data[tengah])
 bawah = tengah - 1;
 else
 atas = tengah + 1;
 }
 return found;
 }

 public static void main(String[] args) {
 Scanner sc = new Scanner(System.in);

 int[] bilangan = {7, 11, 15, 18, 21, 24, 27, 32};

 System.out.print("Bilangan yang dicari: ");
 int bilCari = sc.nextInt();

 boolean ditemukan = binarySearch(bilangan, bilCari);
 if(ditemukan)
 System.out.println("Data ditemukan.");
 else
 System.out.println("Data tidak ditemukan.");
 }
}
```

Solusi bahasa Python.

```
def binarySearch(n, data, cari):
 found = False
 atas = 0
 bawah = n-1
 while(atas <= bawah and not found):
 tengah = (atas + bawah) // 2
 if(data[tengah] == cari):
 found = true
 elif(cari < data[tengah]):
 bawah = tengah - 1
 else:
 atas = tengah + 1
 return found

def main():
 n = 8
 bilangan = [7, 11, 15, 18, 21, 24, 27, 32]
 bilCari = int(input("Bilangan yang dicari:"))

 ditemukan = binarySearch(n, bilangan, bilCari)
 if(ditemukan):
 print("Data ditemukan.")
 else:
 print("Data tidak ditemukan.")

if __name__ == '__main__':
 main()
```

***

Algoritma Pengurutan

Pengurutan merupakan proses untuk menyusun sekumpulan data sehingga dapat terurut dari besar ke kecil atau dari kecil ke besar, dari A ke Z atau dari Z ke A. Banyak metode yang dapat digunakan untuk mengurutkan data tersebut. Pemilihan metode yang akan dipakai dapat dilihat berdasarkan banyak data, struktur data, struktur file dan lain sebagainya. Algoritma pengurutan yang akan dibahas merupakan metode pengurutan dasar yang akan diimplementasikan pada larik (array). Metode ini banyak digunakan untuk data yang tidak begitu besar atau data besar yang sebagian besar datanya sudah terurut.

***

Selection Sort

Metode pengurutan ini merupakan salah satu metode pengurutan yang paling sederhana, dimana metode ini akan mencari angka terbesar/terkecil untuk ditempatkan pada tempat yang seharusnya. Pengurutan secara menaik (kecil ke besar) dengan Selection Sort dikenal dengan Minimum Sort, sedangan pengurutan secara menurun (besar ke kecil) dikenal dengan Maximum Sort.

Pada Minimum Sort, algoritma akan berjalan untuk menemukan nilai element terkecil pada larik (array) kemudian ditukar dengan element pada indeks paling pertama. Kemudian dilanjutkan dengan mencari nilai element terkecil kedua untuk ditukar dengan element pada indeks kedua. Proses akan berjalan sebanyak (N-1) proses.

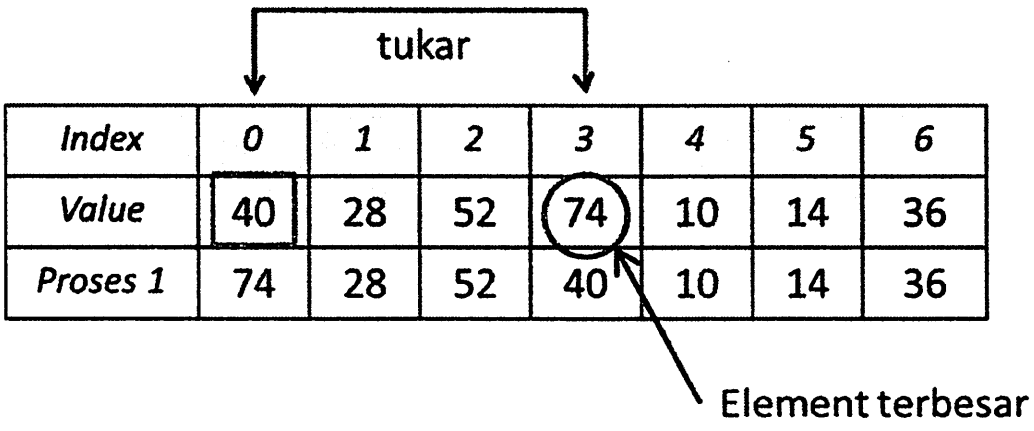
Pada Maximum Sort, algoritma akan berjalan untuk menemukan nilai element terbesar pada larik (array) kemudian ditukar dengan element pada indeks paling pertama. Kemudian dilanjutkan dengan mencari nilai element terbesar kedua untuk ditukar dengan element pada indeks kedua. Proses akan berjalan sebanyak (N-1) proses.

Contoh untuk pengurutan menurun (Maximum Sort):

Isi Array sebelum diurutkan.

Index	*0*	*1*	*2*	*3*	*4*	*5*	*6*
Value	40	28	52	74	10	14	36

Pada proses ini, kita akan mencari indeks element terbesar mulai dari indeks 0 (nol) sampai indeks (N-1), kemudian tukarkan element terbesar tersebut dengan element pada indeks 0 (nol).



Index	*0*	*1*	*2*	*3*	*4*	*5*	*6*
Value	40	28	52	74	10	14	36
Proses 1	74	28	52	40	10	14	36

Setelah proses 1 selesai, maka lanjutkan ke proses 2. Pada proses 2 ini, kita akan mencari indeks element kedua terbesar dari indeks 1 sampai indeks (N-1). Pencarian dimulai dari indeks 1 karena indeks 0 sudah terisi oleh data nilai yang terbesar sehingga dinyatakan sudah pada posisi yang benar. Proses akan dilanjutkan sampai pada proses ke (N-1).

Index	0	1	2	3	4	5	6
Value	40	28	52	74	10	14	36
Proses 1	74	28	52	40	10	14	36
Proses 2	74	52	28	40	10	14	36
Proses 3	74	52	40	28	10	14	36
Proses 4	74	52	40	36	10	14	28
Proses 5	74	52	40	36	28	14	10
Proses 6	74	52	40	36	28	14	10

Berikut adalah solusi dalam bahasa C#.

```
using System;

class Program
{
 static int[] bil = {40, 28, 52, 74, 10, 14, 36};

 static bool MaximumSort()
 {
 for(int proses = 0; proses < bil.Length; ++proses)
 {
 int iMax = proses;
 for(int i = proses + 1; i < bil.Length; ++i)
 if(bil[i] > bil[iMax])
 iMax = i;

 int temp = bil[proses];
 bil[proses] = arr[iMax];
 bil[iMax] = temp;
 }
 }

 static void Main(string[] args)
 {
 Console.WriteLine("Data sebelum diurutkan: " + string.Join(",", bil));
 MaximumSort();
 Console.WriteLine("Data setelah diurutkan: " + string.Join(",", bil));
 }
}
```

Solusi dalam bahasa JAVA.

```
import java.util.Scanner;

public class Program {
 public static int[] bil = {40, 28, 52, 74, 10, 14, 36};

 public static void maximumSort() {
 for(int proses = 0; proses < bil.length; ++proses) {
```

```
 int iMax = proses;
 for(int i = proses + 1; i < bil.length; ++i)
 if(bil[i] > bil[iMax])
 iMax = i;

 int temp = bil[proses];
 bil[proses] = bil[iMax];
 bil[iMax] = temp;
 }
 }

 public static void main(String[] args) {
 System.out.println("Data sebelum diurutkan: " + Array.toString(bil));
 maximumSort();
 System.out.println("Data setelah diurutkan: " + Array.toString(bil));
 }
}
```

Solusi dalam bahasa Python.

```
def maximumSort():
 for proses in range(0, n-1, 1):
 iMax = proses
 for i in range(proses + 1, n, 1):
 if(bil[i] > bil[iMax]):
 iMax = i

 temp = bil[proses]
 bil[proses] = bil[iMax]
 bil[iMax] = temp
 return

def main():
 print("Data sebelum diurutkan:", bil)
 maximumSort()
 print("Data setelah diurutkan:", bil)

if __name__ == '__main__':
 n = 7
 bil = [40, 28, 52, 74, 10, 14, 36]
 main()
```

***

Bubble Sort

Metode pengurutan ini akan menggelembungkan data yang terbesar/terkecil pada setiap prosesnya dengan membandingkan antar element yang bersebelahan. Berikut ini ilustrasi dari proses pengurutan dengan Bubble Sort secara menaik (besar ke kecil).

Bubble sort memiliki proses sebanyak (N-1), berikut ini ilustrasi dari pengurutan pada proses 1. Pengecekan akan dilakukan mulai dari element indeks (N-1) t. Bandingkan dengan element pada indeks sebelumnya (indeks t-1). Jika element pada indeks t lebih kecil, maka lakukan pertukaran dengan element pada indeks t-1. Lakukan pengecekan sampai t-1 adalah indeks nol.

Index	*0*	*1*	*2*	*3*	*4*	*5*	*6*
Value	40	28	52	74	10	14	36
	40	28	52	74	10	14	36
	40	28	52	74	10	14	36
	40	28	52	10	74	14	36
	40	28	10	52	74	14	36
	40	10	28	52	74	14	36
	10	40	28	52	74	14	36

Pada proses 1, pengecekan akan dilakukan hingga indeks nol, sedangkan pada proses 2 pengecekan akan dilakukan hingga indeks 1. Hal ini dilakukan, karena pada proses pertama kita telah mendapatkan nilai terkecil dari deretan data yang ada pada larik (array) sehingga tidak perlu dilakukan pengecekan kembali. Sama hal-nya pada proses 3, pengecekan akan dilakukan hingga indeks 2, dikarenakan indeks nol dan 1 sudah terisi oleh element terkecil pertama dan element terkecil kedua.

Index	*0*	*1*	*2*	*3*	*4*	*5*	*6*
Value	40	28	52	74	10	14	36
Proses 1	10	40	28	52	74	14	36
Proses 2	10	14	40	28	52	74	36
Proses 3	10	14	28	40	36	52	74
Proses 4	10	14	28	36	40	52	74
Proses 5	10	14	28	36	40	52	74
Proses 6	10	14	28	36	40	52	74

Berikut adalah solusi dalam bahasa C#.

```
using System;

class Program
{
 static int[] bil = {40, 28, 52, 74, 10, 14, 36};

 static bool BubbleSort()
 {
 for(int proses = 0; proses < bil.Length; ++proses)
 {
 for(int t = proses + 1; t < bil.Length; ++t)
 if(bil[t] > bil[t-1])
 {
 int temp = bil[proses];
 bil[proses] = bil[iMax];
 bil[iMax] = temp;
 }
 }
 }

 static void Main(string[] args)
 {
 Console.WriteLine("Data sebelum diurutkan: " + string.Join(",", bil));
 BubbleSort();
 Console.WriteLine("Data setelah diurutkan: " + string.Join(",", bil));
 }
}
```

Solusi dalam bahasa JAVA.

```
import java.util.Scanner;

public class Program {
 public static int[] bil = {40, 28, 52, 74, 10, 14, 36};

 public static void bubbleSort() {
 for(int proses = 0; proses < bil.length; ++proses) {
 for(int t = proses + 1; i < bil.length; ++i)
 if(bil[t] > bil[t-1])
```

```
 {
 int temp = bil[proses];
 bil[proses] = bil[iMax];
 bil[iMax] = temp;
 }
 }
 }

 public static void main(String[] args) {
 System.out.println("Data sebelum diurutkan: " + Array.toString(bil));
 bubbleSort();
 System.out.println("Data setelah diurutkan: " + Array.toString(bil));
 }
}
```

Solusi dalam bahasa Python.

```
def bubbleSort():
 for proses in range(0, n-1, 1):
 for t in range(n-1, proses, -1):
 if(bil[t] < bil[t-1]):
 temp = bil[proses]
 bil[proses] = bil[iMax]
 bil[iMax] = temp
 return

def main():
 print("Data sebelum diurutkan:", bil)
 bubbleSort()
 print("Data setelah diurutkan:", bil)

if __name__ == '__main__':
 n = 7
 bil = [40, 28, 52, 74, 10, 14, 36]
 main()
```

***

Insertion Sort

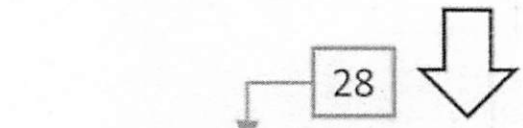
Metode pengurutan ini mirip dengan metode pengurutan kartu. Insertion Sort mengurutkan dengan mengambil data satu per satu dan disisipkan pada tempat yang seharusnya. Pengurutan dimulai dari element ke-2 sampai dengan data terakhir, jika ditemukan data yang lebih kecil, maka akan ditempatkan diposisi yang seharusnya. Pada saat disisipkan, element-element yang lain akan bergeser ke belakang. Insertion sort mempunyai waktu penyelesaian lebih cepat dibandingkan dengan Selection Sort dan Bubble Sort.

Berikut ini ilustrasi dari pengurutan data dengan menggunakan Insertion Sort secara menaik (kecil ke besar). Metode ini akan memilki proses sebanyak (N-1).

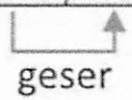
Value	40	28	52	74	10	14	36
Index	*0*	*1*	*2*	*3*	*4*	*5*	*6*

Langkah-langkah pengurutan insertion sort.

Value	40	28	52	74	10	14	36
Index	*0*	*1*	*2*	*3*	*4*	*5*	*6*



Value	40		52	74	10	14	36
Index	*0*	*1*	*2*	*3*	*4*	*5*	*6*



Index	0	1	2	3	4	5	6
Value	40	28	52	74	10	14	36
Proses 1	28	40	52	74	10	14	36
Proses 2	28	40	52	74	10	14	36
Proses 3	28	40	52	74	10	14	36
Proses 4	10	28	40	52	74	14	36
Proses 5	10	14	28	40	52	74	36
Proses 6	10	14	28	36	40	52	74

Berikut adalah solusi dalam bahasa C#.

```
using System;

class Program
{
 static int[] bil = {40, 28, 52, 74, 10, 14, 36};

 static bool InsertionSort()
 {
 for(int proses = 0; proses < bil.Length; ++proses)
 {
 int temp = bil[proses];
 int i = proses - 1;
 while(temp < bil[i] && i > 0)
 {
 bil[i + 1] = bil[i];
 i = i - 1;
 }
 if(temp >= bil[i])
 {
 bil[i + 1] = temp;
 }
 else
 {
 bil[i + 1] = bil[i];
 bil[i] = temp;
 }
 }
 }

 static void Main(string[] args)
 {
 Console.WriteLine("Data sebelum diurutkan: " + string.Join(",", bil));
 InsertionSort();
 Console.WriteLine("Data setelah diurutkan: " + string.Join(",", bil));
 }
}
```

Solusi dalam bahasa JAVA.

```
import java.util.Scanner;

public class Program {
 public static int[] bil = {40, 28, 52, 74, 10, 14, 36};

 public static void insertionSort() {
 for(int proses = 1; proses < bil.length; ++proses) {
 int temp = bil[proses];
 int i = proses - 1;
 while(temp < bil[i] && i > 0) {
 bil[i + 1] = bil[i];
 i = i - 1;
 }
 if(temp >= bil[i]) {
 bil[i + 1] = temp;
 } else {
 bil[i + 1] = bil[i];
 bil[i] = temp;
 }
 }
 }

 public static void main(String[] args) {
 System.out.println("Data sebelum diurutkan: " + Array.toString(bil));
 insertionSort();
 System.out.println("Data setelah diurutkan: " + Array.toString(bil));
 }
}
```

Solusi dalam bahasa Python.

```
def insertionSort():
 for proses in range(1, n, 1):
 temp = bil[proses]
 i = proses - 1
 while(temp < bil[i] and i > 0):
 bil[i+1] = bil[i]
 i = i - 1
 if(temp >= bil[i]):
 bil[i + 1] = temp
 else:
 bil[i + 1] = bil[i]
 bil[i] = temp
 return

def main():
 print("Data sebelum diurutkan:", bil)
 insertionSort()
 print("Data setelah diurutkan:", bil)

if __name__ == '__main__':
 n = 7
 bil = [40, 28, 52, 74, 10, 14, 36]
 main()
```

***