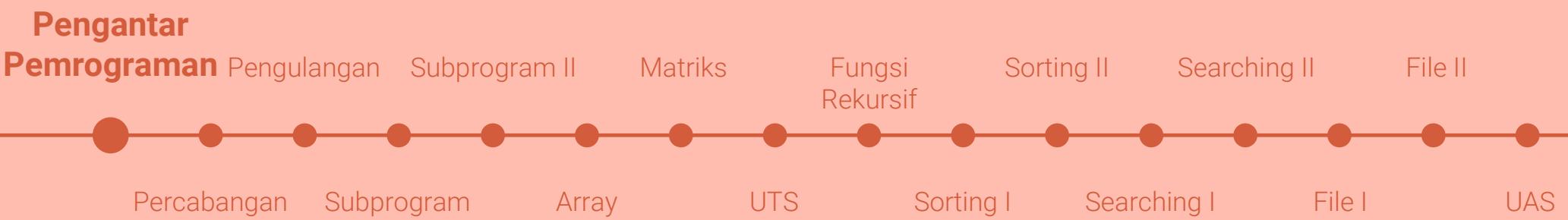


# DASAR PEMROGRAMAN

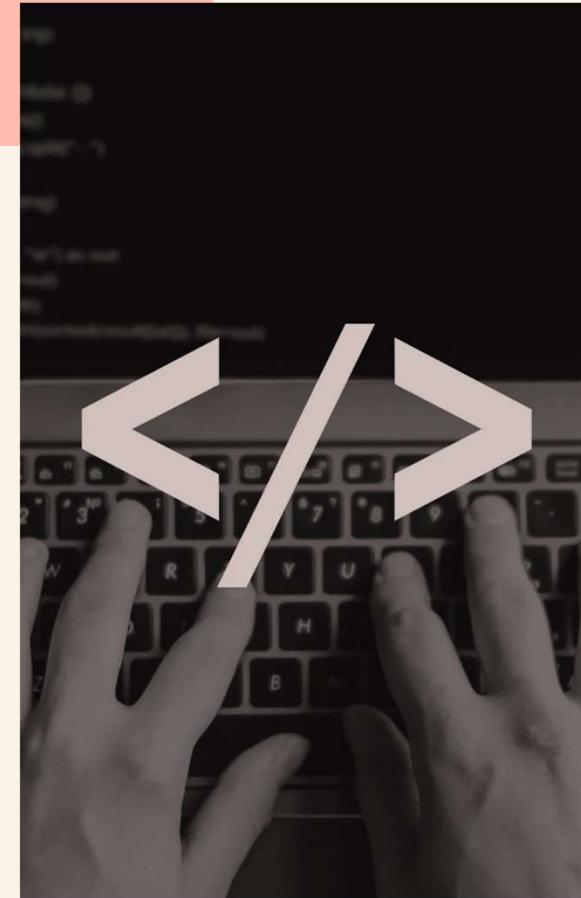
# Pertemuan I





# Tujuan

- Mampu menjelaskan bagaimana proses dari source code menjadi program yang dapat dieksekusi.
- Mengetahui IDE dan fungsinya dalam pengembangan program.
- Paham struktur dari sebuah program dalam bahasa pemrograman tertentu.
- Paham berbagai jenis tipe data primitif beserta kegunaannya.
- Paham dan dapat menggunakan variable, type, konstanta, input/output, dan sekuens.
- Mampu memahami persoalan yang memberikan solusi dalam bentuk program sederhana dengan memanfaatkan variable, type, konstanta, ekspresi dasar, input/output, dan sekuens.





# Materi

PERAN BAHASA PEMROGRAMAN

SEJARAH BAHASA PEMROGRAMAN

KLASIFIKASI BAHASA PEMROGRAMAN

PARADIGMA PEMROGRAMAN

VARIABEL, KONSTANTA, DAN TIPE DATA

OPERASI

INPUT DAN OUTPUT

DECLARATION, ASSIGNMENT, DAN EXPRESSION

RUNTUNAN

# PERAN BAHASA PEMROGRAMAN

---

भारत में  
आपका  
स्वागत है



???



## International Morse Code

1. A dash is equal to three dots.
2. The space between parts of the same letter is equal to one dot.
3. The space between two letters is equal to three dots.
4. The space between two words is equal to seven dots.

A		U	
B		V	
C		W	
D		X	
E		Y	
F		Z	
G			
H			
I			
J			
K		1	
L		2	
M		3	
N		4	
O		5	
P		6	
Q		7	
R		8	
S		9	
T		0	

Code	Mnemonic	Function	Description	Possible flags
0000	ILL	(noop)	Illegal opcode	E
0001	NOTA	NOT A	Bitwise complement of A	Z, N
0010	ILL	(noop)	Illegal opcode	E
0011	ILL	(noop)	Illegal opcode	E
0100	SUBB	A-B	Subtracts B from A	Z, N, C
0101	NEGA	-A	2's complement of A	Z, N, C
0110	ILL	(noop)	Illegal opcode	E
0111	AND	A AND B	Bitwise AND of AB	Z
1000	ADD	A+B	Addition of A and B	Z, N, C
1001	NOTB	NOT B	Bitwise complement of B	Z, N
1010	ILL	(noop)	Illegal opcode	E
1011	ILL	(noop)	Illegal opcode	E
1100	SUBA	B-A	Subtracts A from B	Z, N, C
1101	NEGB	-B	2's complement of B	Z, N, C
1110	ILL	(noop)	Illegal opcode	E
1111	OR	A OR B	Bitwise OR of A—B	Z

Code	Mnemonic	Function	Description
0000	ILL	(noop)	Illegal opcode
0001	NOTA	NOT A	Bitwise NOT of A
0010	ILL	(noop)	Illegal opcode
0011	ILL	(noop)	Illegal opcode
0100	SUBB	A-B	Subtract B from A
0101	NEGA	-A	2's complement of A
0110	ILL	(noop)	Illegal opcode
0111	AND	A AND B	Bitwise AND of AB
1000	ADD	A+B	Addition of A and B
1001	NOTB	NOT B	Bitwise complement of B
1010	ILL	(noop)	Illegal opcode
1011	ILL	(noop)	Illegal opcode
1100	SUBA	B-A	Subtracts A from B
1101	NEG B	-B	2's complement of B
1110	ILL	(noop)	Illegal opcode
1111	OR	A OR B	Bitwise OR of A-B

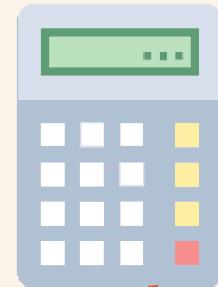


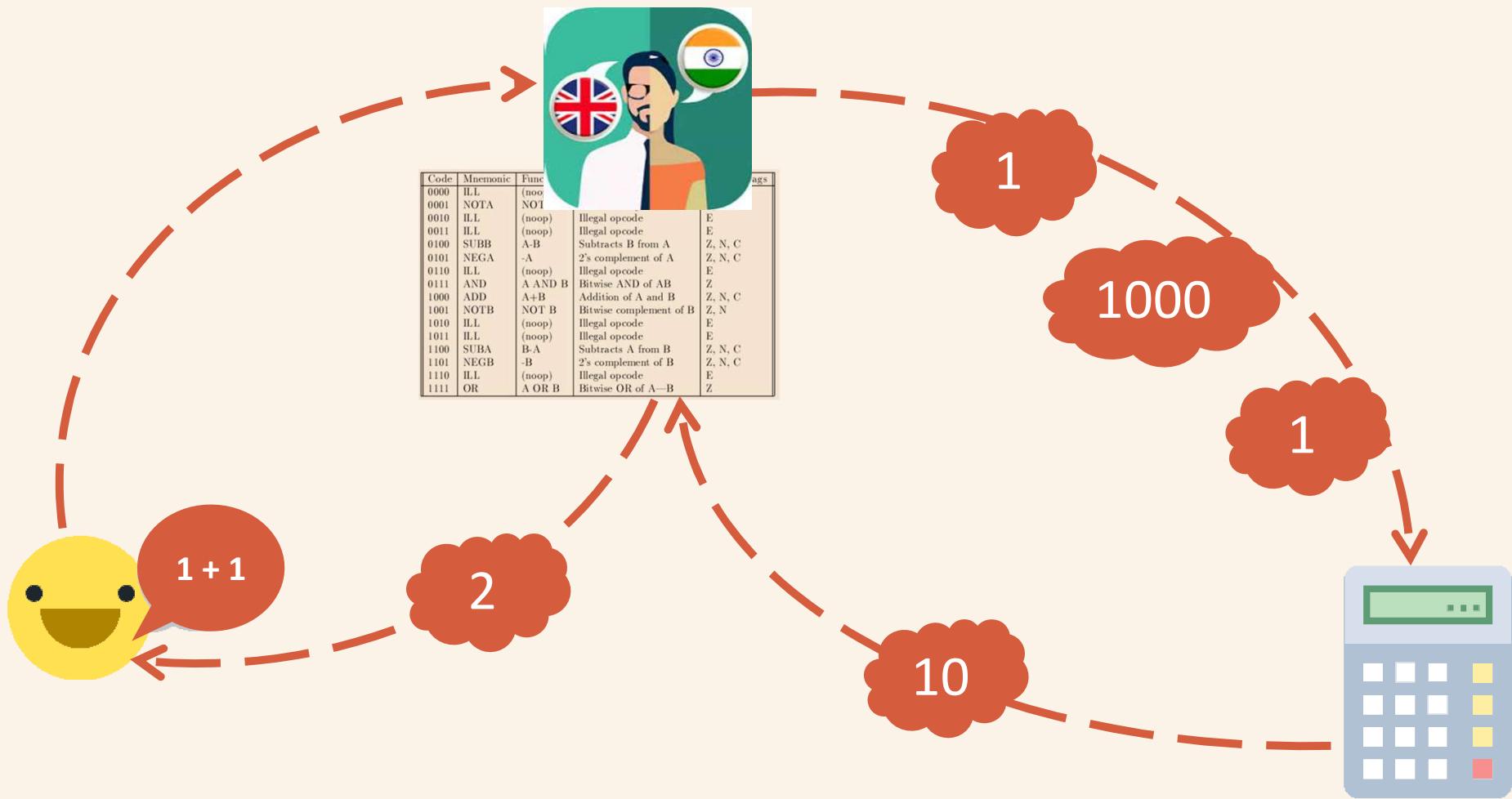
1

1000

1

10





# SEJARAH BAHASA PEMROGRAMAN

---

# 86-67 SM

- Alat dan mesin hitung sudah ada sejak 1000 tahun lalu.
- Peralatan ini membantu melakukan perhitungan tertentu.
- Antikythera, kalkulator mekanik untuk mengkalkulasi posisi matahari, bulan, dan mungkin beberapa planet. Hal ini memungkinkan penggunaan perangkat sebagai kalender.

<https://youtu.be/EZy4a5uTYH0?si=YwPfismxd6YrAkHL>

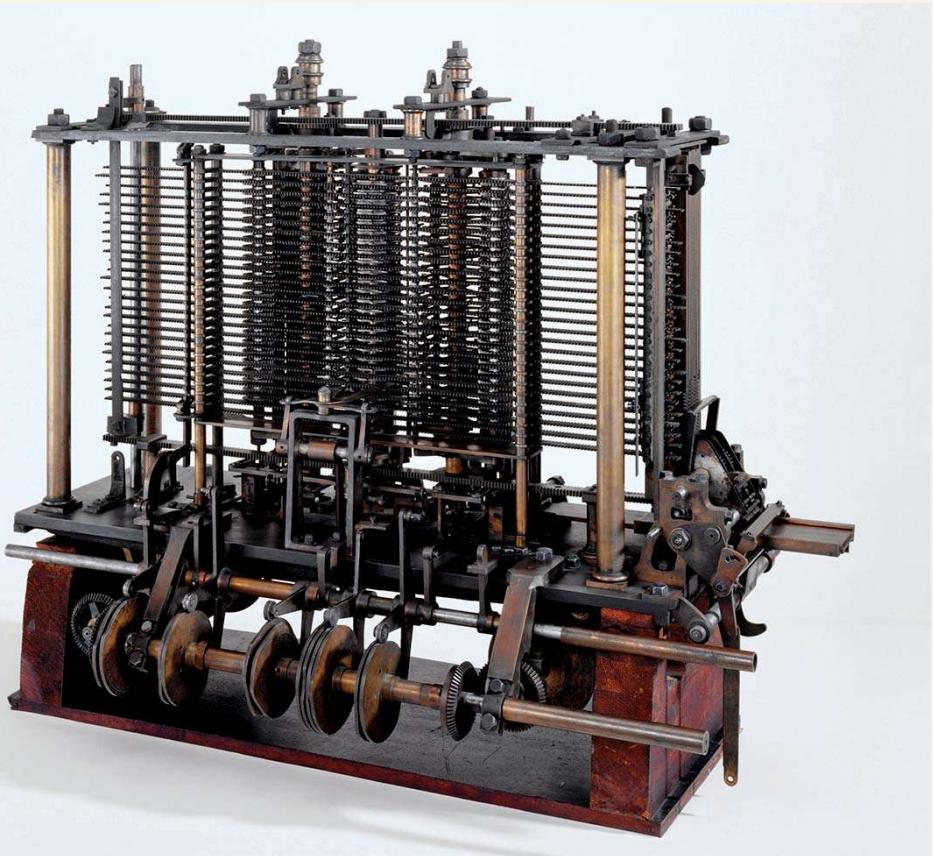


# 1801



- Alat tenun otomatis yang bisa membuat pola-pola rumit di kain sutra didesain oleh Joseph Marie Jacquard
- Pola pada mesin tenun Jacquard Loom dikendalikan oleh rangkaian kartu berlubang (*punch card*).
- Kemampuan untuk menyimpan dan secara otomatis mereproduksi operasi yang kompleks diaplikasikan secara luas dalam pembuatan tekstil.

<https://youtu.be/MQzpLLhN0fY?si=waCan-6ZxZUTXhma>



# 1837

- Analytical Engine, komputer mekanik yang didesain oleh ahli matematika Charles Babbage
- Mampu menyimpan data, melakukan operasi aritmatika, dan menjalankan control flow
- Input berupa formula dan data diterima mesin dalam bentuk *punch card*, terinsipirasi dari mesin tenun Jacquard.

[https://youtu.be/eMy4vSZ-J\\_I?si=MAgxts6L5Brv\\_yzp](https://youtu.be/eMy4vSZ-J_I?si=MAgxts6L5Brv_yzp)



# 1843

"The Analytical Engine does not occupy common ground with mere "calculating machines." In enabling mechanism to combine together general symbols, in successions of unlimited variety and extent, a uniting link is established between the operations of matter and the abstract mental processes of the most abstract branch of mathematical science. A new, a vast and powerful language is developed"

Ada Lovelace, 1843

<https://youtu.be/fEmKuJ8XL68?si=piSMwo1NgHKfa11S>



Sejak itu, komputer dan Internet telah membuat pemrograman lebih dari sekedar memecahkan masalah matematika

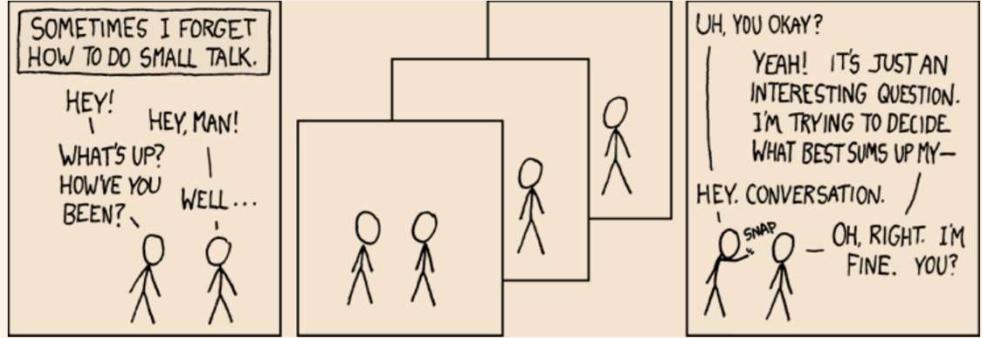
- *Code* sekarang memengaruhi hampir setiap aspek kehidupan kita.
- Sistem jaringan yang kompleks.
- *Code* digunakan dalam keselamatan dan keamanan.
- Berbagai jenis perangkat, sistem operasi, dan bahasa pemrograman.

# BAHASA PEMROGRAMAN

---

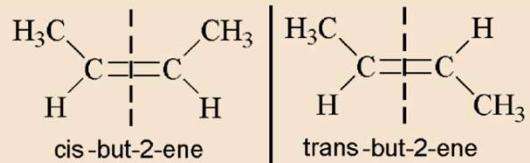
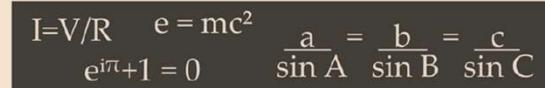
”

Programming languages are  
**formal languages** that have been  
designed to express computations.



# NATURAL LANGUAGE

- Contoh: Bahasa Inggris, Arab, Mandarin, Jawa, dsb
  - Berevolusi, tidak didesain
  - Kamus dan tata bahasa sebagai usaha memformalkan
  - Ambigu, redundan

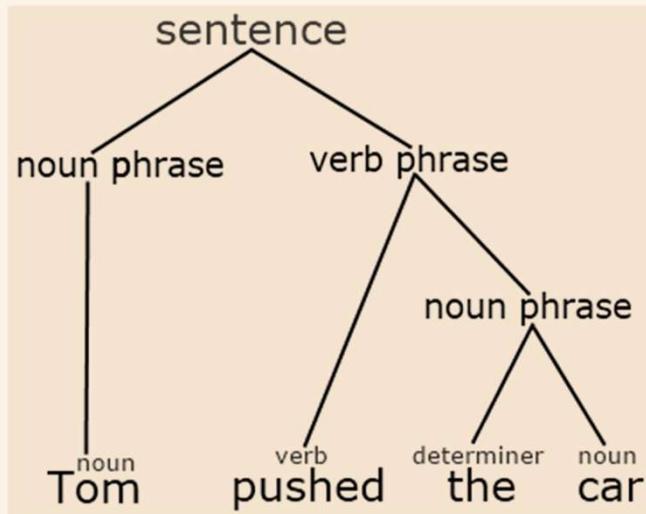


```
if(age >= 18)
{
    System.out.println("Person is 18 or over");
}
else if(age >= 16)
{
    System.out.println("Person is 16 or over");
}
```

# FORMAL LANGUAGE

- Contoh: Bahasa pemrograman, notasi kimia dan matematika
  - Didesain untuk tujuan tertentu
  - Aturan penulisan yang ketat
  - Harfiah

# SINTAKS DAN PARSING



- Program dibuat dari rangkaian pernyataan (*statement*).
- Sintaks
  - Aturan yang mengatur bagaimana suatu pernyataan dituliskan.
  - Harus menggunakan struktur dan token yang benar.
- Parsing
  - Proses menguraikan pernyataan untuk memahami maknanya.
  - Komputer menguraikan code yang dituliskan.
- Syntax error akan ditampilkan jika pernyataan tidak dituliskan dengan benar.

# **AMBIGUITAS**

Ambiguitas harus dihindari supaya komputer bisa dengan mudah memahami apa yang harus dilakukan dan bagaimana caranya

# **REDUNDANSI**

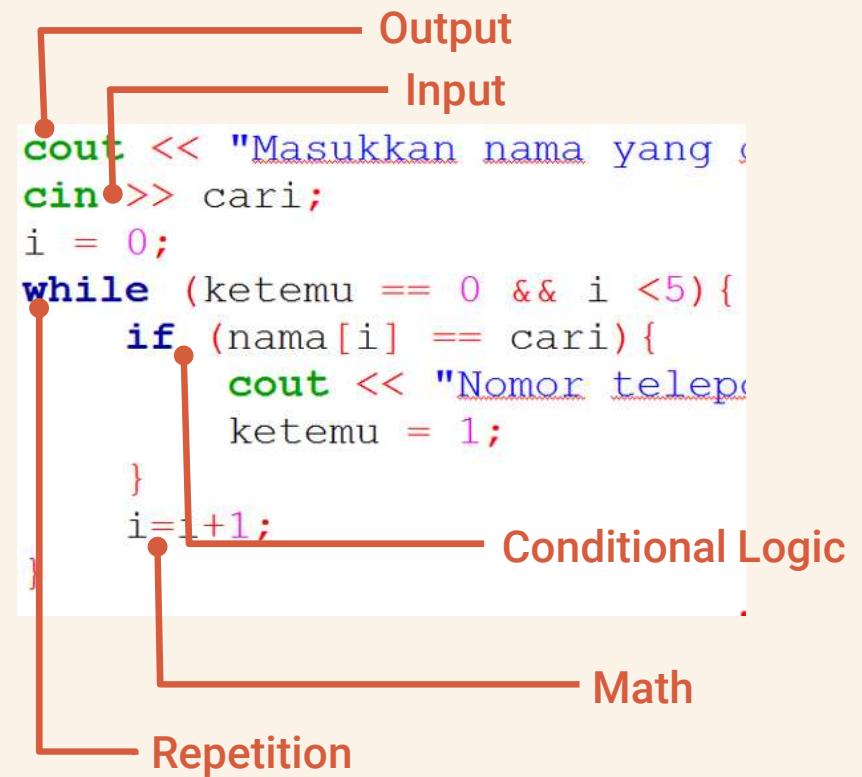
Redundansi harus dihindari agar pernyataan mudah untuk diuraikan dan murah dimengerti oleh manusia dan komputer.

” ”

Programming languages are  
formal languages that have been  
designed to **express computations**

# MENYATAKAN PERHITUNGAN

- **Input:** menerima data dari *keyboard*, file, atau perangkat lain.
- **Output:** Menampilkan data ke layar atau menuliskan data ke file atau perangkat lain.
- **Maths:** Melakukan operasi dasar matematika seperti penambahan dan perkalian.
- **Conditional Logic:** Memeriksa suatu kondisi dan menjalankan *code* yang sesuai.
- **Repetition:** Melakukan suatu aksi berulang-ulang, biasanya dengan beberapa variasi.



# **KLASIFIKASI BAHASA PEMROGRAMAN**

---

**Human Language**

**High Level  
Programming Language**

**Low Level  
Programming Language**

**Machine Language**

**Computer Hardware**

Bahasa Indonesia, Arab, Inggris

Phyton, Java, C++

```
int f(const int n) {  
    return (n % 2 == 0) ? 3 * n + 1 : 4 * n - 3;  
}
```

Assembly

```
mov    %edi, %eax
```

Hexadecimal

```
89 F8 A9 01 00 00 00 75 06 6B C0 03 FF
```



**HIGH LEVEL**

**LOW LEVEL**

## Human Language

## High Level Programming Language

- Sintaks mudah dipahami manusia
- *Portable* untuk berbagai tipe komputer
- Komputer biasanya menerjemahkan ke low-level code untuk kita

## Low Level Programming Language

- Sintaks dipahami oleh computer tipe tertentu
- Sudah jarang digunakan saat ini

## Machine Language

## Computer Hardware



HIGH LEVEL

LOW LEVEL

# METODE PENTERJEMAHAN

Compile

write a  
program

compile  
it

distribute  
it

users  
run it

## COMPILE

Menerjemahkan seluruh program menjadi *executable*.

Interpreted

write a  
script

distribute  
it

users  
run it

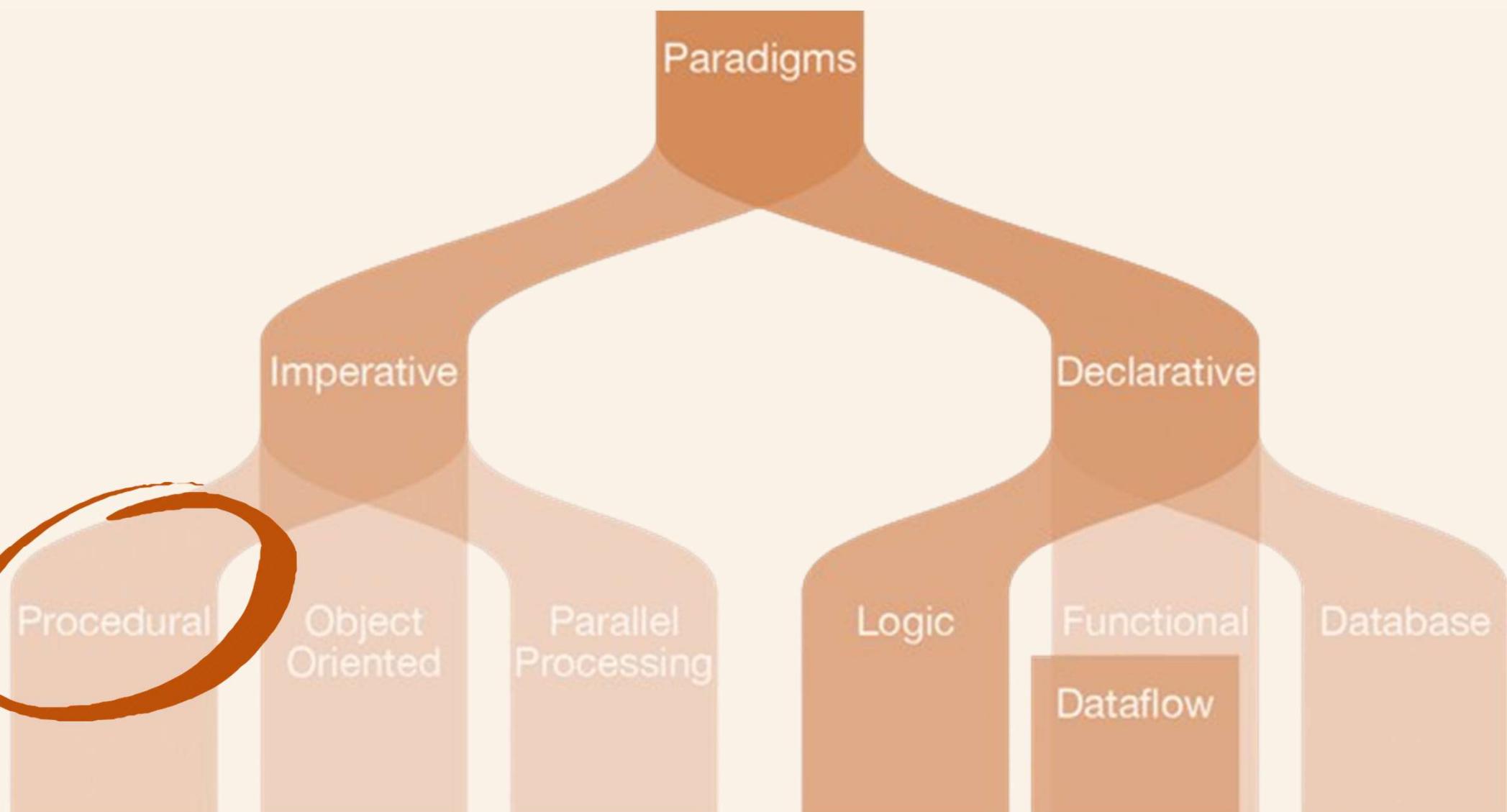
each line  
is interpreted  
as it's run

## INTERPRETATION

Membaca perbaris, mengartikan (*parse*) baris yang dibaca, menjalankan komputasi.

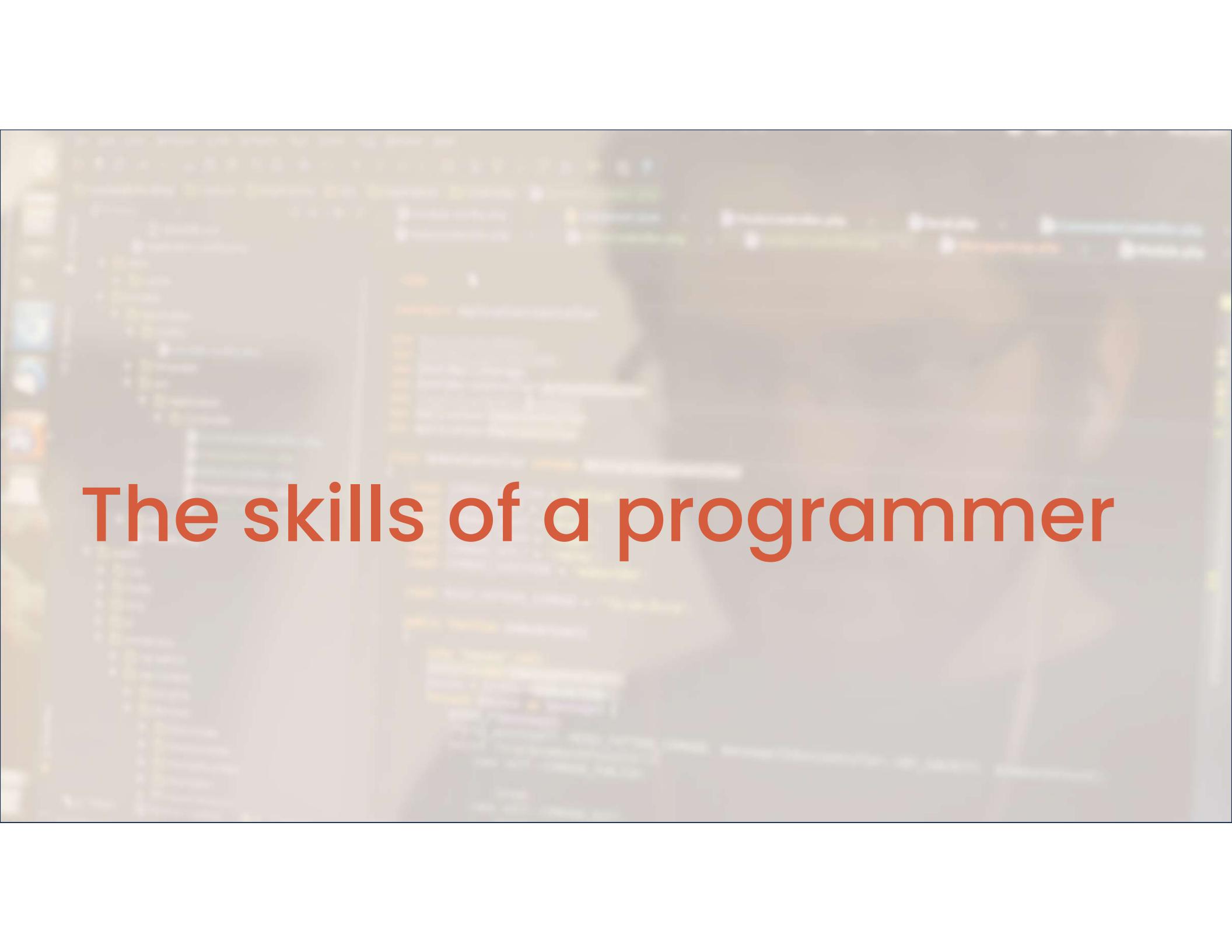
# PARADIGMA PEMROGRAMAN

---



# PEMROGRAMAN PROSEDURAL

- Pendekatan top-down
- Menuliskan perintah Langkah demi Langkah untuk memberitahu komputer apa yang harus dilakukan.
- Perintah atau aksi dijalankan secara berurutan (sekuensial).
- Setiap aksi akan memberikan efek eksekusi tertentu.



The skills of a programmer

Theory + Practice



---

# Clarity and Simplicity

- Code is often return to at later date
- Or inherited by other programmers
- Even simple programs get complex very quickly, so
- Keep code neat and easy to understand
- Use sensible approaches to naming
- Provide comments and instructions
- Modularise code to simplify use

# BAHASA PEMROGRAMAN C++

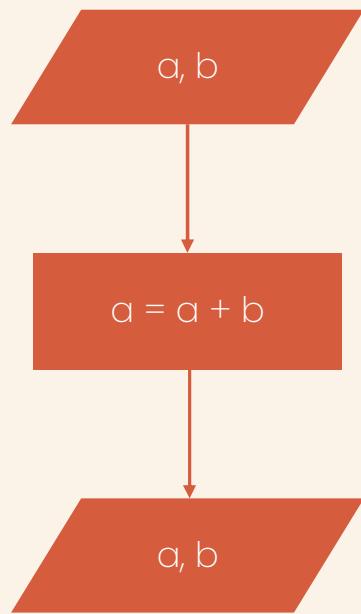
---



# C++

- C++ merupakan bahasa pemrograman general purpose dan multi paradigma (procedural, object oriented)
- Bahasa pemrograman yang sangat popular dan banyak digunakan.
- Dikembangkan oleh Bjarne Stroustrup mulai tahun 1979 di Bell Labs
- Merupakan pengembangan dari Bahasa C (prosedural murni) dengan penambahan konsep, object-orientation
- Dalam kuliah ini, hanya akan menggunakan paradigma procedural
- Merupakan bahasa yang case sensitive → perbedaan huruf besar dan kecil berpengaruh

## FLOWCHART



## PSEUDOCODE

```
read a  
read b  
  
a = a + b  
  
print a  
print b
```

## SOURCE CODE

```
#include <iostream>  
  
using namespace std;  
  
int main()  
{  
    cin >> a;  
    cin >> b;  
  
    a = a + b;  
  
    cout << a;  
    cout << b;  
  
    return 0;  
}
```

# **STRUKTUR DASAR PEMROGRAMAN**

---

## **Declaration of Pre-processor Statements**

Berisi *library function* dan file *header*

## **Global Declaration**

Semua nama *function* dan variable yang ingin dikenali disemua bagian program  
(Dibahas di pertemuan 4 )

## **Main Function**

Bagian yang pertama kali akan dijalankan program

## **User Defined Function**

Bagian yang berisi realisasi *function*  
(Dibahas di pertemuan 4)

```
#include <iostream>
using namespace std;
```

Declaration of Pre-processor Statements

```
const int seribu = 1000;
int ubah (int angka);
```

Global Declaration

```
int main()
{
    int nilai;
    cout << "Masukkan angka yang akan dikonversi: ";
    cin >> nilai;
    cout << "Nilai dibagi seribu adalah " << ubah(nilai) << endl;
    cout << "Nilai dikali seribu adalah " << nilai * seribu;

    return 0;
}
```

Main Function

```
int ubah (int angka)
{
    return (angka/seribu);
}
```

User Defined Function

# PREPROCESSOR STATEMENT



- **iostream** adalah salah satu header file yang ada di C++. Header ini digunakan untuk fungsi input dan output yang ada di C++.  
Contoh fungsi input/output: **cin** dan **cout**
- **using namespace std** adalah perintah yang digunakan untuk memberitahukan kepada compiler C++ bahwa kita akan menggunakan semua fungsi/class/file yang terdapat dalam namespace std
- **namespace** adalah awalan yang digunakan untuk semua kumpulan fungsi tertentu

# MAIN FUNCTION



- Program C++ terdiri dari satu atau lebih fungsi.
- **Main** adalah sebuah fungsi, dan hanya boleh terdapat satu fungsi main dalam setiap program C++.
- Setiap program akan memulai dieksekusi pada fungsi main ini.

# **VARIABLE, KONSTANTA, DAN TIPE DATA**

---

$$c + 3 = M$$

$$1 + 3 = 4$$

$$2 + 3 = 5$$

C	1	2		
m	4	5		

- **Variable** adalah suatu wadah dengan nama tertentu untuk menyimpan suatu nilai.
- Dalam pemrograman C++, perintah assignment digunakan untuk memberikan nilai terhadap suatu variable.

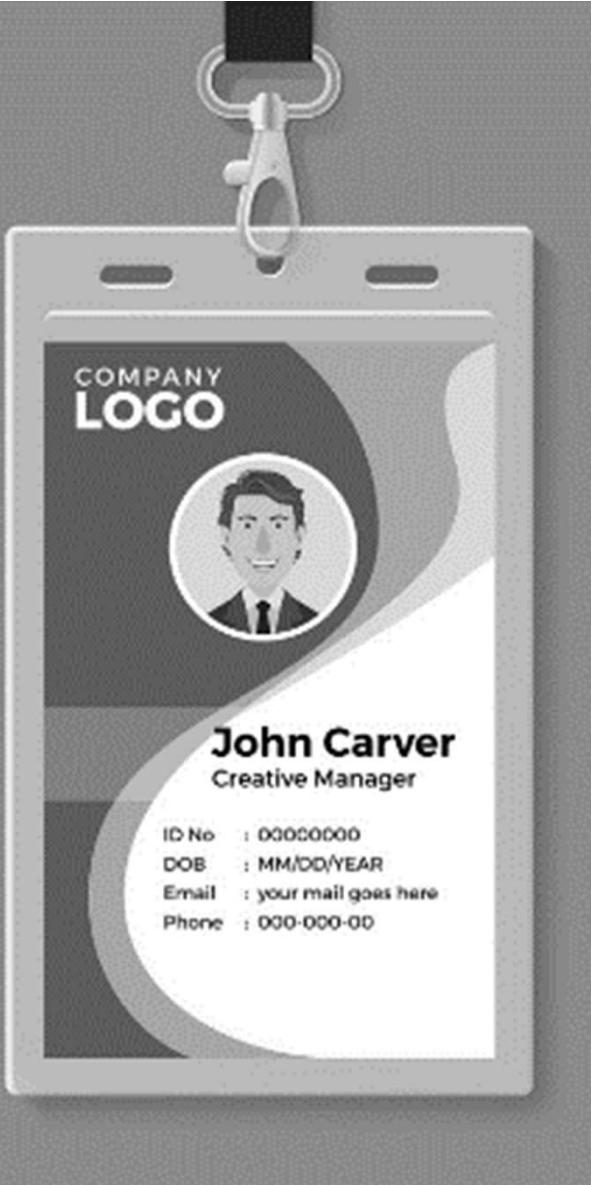
- 
- **Value** adalah suatu data yang disimpan oleh variable tertentu.

$g = 9.8$

$\phi i = 3.14$

$2x + 4y - 9$

- **Konstanta** adalah suatu wadah dengan nama tertentu untuk menyimpan suatu nilai yang tidak bisa diubah.



Setiap data memiliki jenis yang berbeda-beda

- Data nama berbeda dengan nomor telepon
  - Data nomor telepon dibentuk dari kumpulan angka
  - Data nama dibentuk dari serangkaian huruf
- Untuk setiap jenis data juga memiliki memiliki **rentang** yang berbeda
  - Data tanggal lahir, panjang hari dan bulan 2 digit tapi memiliki rentang yang berbeda. 1-31 untuk hari, 1-12 untuk bulan.
  - Data nomor telepon memiliki panjang 10 digit

# TIPE DATA

I 0 Boolean (bool)

1 Integer (int)

1.0 Real (float)

A Character (char)

JOHN String (string)

**1 + 1 = 2**

integer

**“1” + “1” = “11”**

string

**‘1’ + ‘1’ = ‘1’**

character

**1.0 + 0.9 = 2.0**

float

# OPERASI

---



# OPERASI

- Operasi terhadap variable sangat tergantung dari tipe datanya
- Operasi perhitungan akan memerlukan operator  $+, -, \div, \times$  untuk melakukan kalkulasi
- Operasi  $“+”$  pada tipe data bukan numerik memiliki arti yang berbeda

$$1 + 1 = \boxed{2}$$

“halo” + “apa kabar” = **haloapa kabar**

# OPERASI UNTUK TIPE DATA

TIPE DATA	JENIS OPERASI	OPERASI
Integer	Aritmatika	/ + - %
	Relational	< > <= >= == !=
Float	Aritmatika	/ + -
	Relational	< > <= >= == !=
String	Relasional	== !=
Character	Relasional	== !=
Boolean	Logika	&&    !
	Relasional	!= ==

# **DECLARATION, ASSIGNMENT, EXPRESSION**

---

```
int main()
{
    int a, b;
    const int seribu = 1000;
    a = 10;
    b = 5;

    a = a + b;
    b = b * seribu;
    b = b - a;

    return 0;
}
```

Dalam pemrograman C++, sebelum melakukan perhitungan, suatu variable harus:

1. Dideklarasikan, agar dikenali oleh program
2. Diinisiasi, diberikan nilai awal agar dapat diproses

```
int a, b;  
const int seribu = 1000;  
a = 10;  
b = 5;
```

```
a = a + b;  
b = b * seribu;  
b = b - a;
```

## Deklarasi variabel

Deklasasi sii lko nusturka  
krisis taintida eun tuuk  
variabel a dan b

## Komputasi/ manipulasi data

```
int a, b;  
const int seribu = 1000;  
a = 10;  
b = 5;
```

**declaration**

**assignment**

```
a = a + b;  
b = b * seribu;  
b = b - a;
```

**expression**

## declaration

```
<tipe data> <nama variable>;  
string no_telepon;
```

## assignment

```
<nama variable> = <value>;  
<nama variable> = <ekspression>;  
a = 100;  
a = b + 100;
```

## expression

```
<nama variable/value> <operasi>  
<nama variable/value>;  
a == 100;  
b + 100;  
1 != "1";
```

# **INPUT DAN OUTPUT**

---



# INPUT

Membaca data dari piranti masukan  
(keyboard).

Syntax :      `cin >> <nama variable>;`  
                  `cin >> angka;`

Perintah **cin** hanya bisa digunakan jika pada  
pada bagian pre-processor statement kita  
mendeklarasikan header file **iostream**



# OUTPUT

Menampilkan data ke peranti keluaran (monitor).

Syntax :      `cout << <nama variable>;`  
                  `cout << angka;`

Perintah **cout** hanya bisa digunakan jika pada pada bagian pre-processor statement kita mendeklarasikan header file **iostream**

```
#include <iostream>

using namespace std;

int main()
{
    int tanggal, bulan, tahun;
    cout << "Hello world!" << endl ;
    cin >> tanggal;
    cin >> bulan;
    cin >> tahun;

    cout << "Hari ini : " << tanggal << "-" << bulan << "-" << tahun" ;

    return 0;
}
```

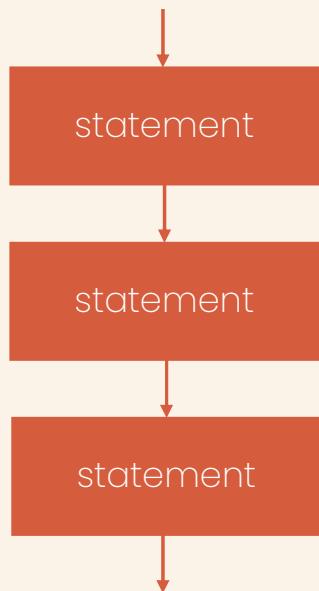
```
C:\Users\mered\Documents\codeblocks\PTIA\test\bin\Debug\test.exe
Hello world!

C:\Users\mered\Documents\co
Hello world!
01
01
C:\Users\mered\Documents\codeblocks\PTIA\
Hello world!
01
01
2020
Hari ini : 1-1-tahun
```

# RUNTUNAN

---

# RUNTUNAN



- Struktur algoritma dan pemrograman yang paling dasar dan sederhana.
- Berisi rangkaian perintah yang diproses secara berurutan atau satu per satu mulai dari perintah pertama hingga perintah terakhir.

# CONTOH

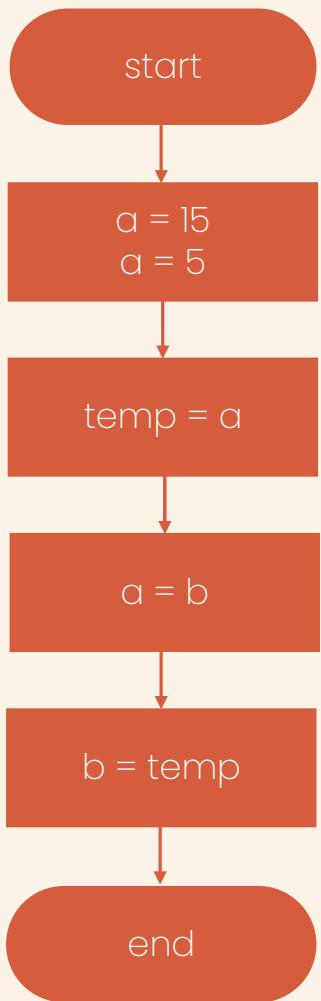
Bagaimana menukar nilai pada dua buah variabel?

$$\begin{array}{l} A = 15 \\ B = 5 \end{array} \quad \rightarrow \quad \begin{array}{l} A = 5 \\ B = 15 \end{array}$$









```
int main()
{
    int temp, a = 15, b = 5;
    temp = a;
    a = b;
    b = temp;

    return 0;
}
```

