

Pemrograman Dasar dan Pengantar Algoritma.

By Abdan Hafidz

Kenalan Dikit

<https://abdanhafidz.com>

Informatics Problem Setter & Tutorial Tutor | [TRY OUT 5 PERSIAPAN OSNK by Alpha Academy]
Alpha Academy · Part-time
Mar 2024 - Present · 2 mos
Remote
Skills: discrete math · Mathematics · logic · Algorithms · Computational Intelligence
 SOAL DAN PEMBAHASAN INFORMATIKA.pdf

Scientific Committee on National Programming Contest (NPC)
Schematics ITS · Seasonal
Mar 2024 - Present · 2 mos
Surabaya, Jawa Timur, Indonesia · Hybrid
Skills: Competitive Programming · Data Structures · Algorithms · Computer Science · Analytic Problem Solving

Informatics Tutor - Pelatihan OSNK 2024
Lembaga Olimpiade Pendidikan Indonesia · Part-time
Mar 2024 - Present · 2 mos
Remote
Teaching National Olympiad in Informatics (NOI) topics : Discrete Math, Algorithmics, Computational Thinking
 Latihan Soal Algoritmitika 2.pdf
 LOPI CT 1.pdf

IT Engineer & Developer (Backend)
Ini Lho ITS! · Seasonal
Oct 2023 - Present · 7 mos
Surabaya, East Java, Indonesia · Hybrid
Skills: GitHub · MySQL · Express JS

Member
GDSC ITS
Oct 2023 - Present · 7 mos
Skills: Web Development · Machine Learning · Artificial Intelligence (AI)

EvaluatorKSN
Part-time · 3 yrs 1 mo

Informatics Teacher
Oct 2023 - Present · 7 mos
Remote
Teaching SMAN 1 Raya Simalungun's Students for preparation of National Science Olympiad (OSN) in Informatics
Skills: Problem Solving · C++ · Programming · discrete math
 PsP Basic.pdf
 Graf dan Tree.pdf
 Day 1 -- LOLOS OSNK INFORMATIKA 2024 AMIIN.pdf

Website Developer
Apr 2021 - Present · 3 yrs 1 mo
Skills: JavaScript · PHP · MySQL

Back End Developer
MABA CUP ITS 2023 · Seasonal
Sep 2023 - Present · 8 mos
Surabaya, Jawa Timur, Indonesia
Skills: Prisma ORM · PostgreSQL · GitHub · Express.js

Member
Google Developers Group
Aug 2023 - Present · 9 mos
Surabaya, East Java, Indonesia

Quantitative Knowledge & Mathematical Reasoning Teacher
Mathan Group · Part-time
Aug 2023 - Present · 9 mos
Remote
Teaching in field of College Entrance examination preparation (SNBT/UTBK)
Skills: Mathematics · Critical Thinking · Calculus · Analytical Reasoning

Chief Executive Officer
Feez Edutech · Full-time
Aug 2022 - Present · 1 yr 9 mos
Jakarta, Jakarta, Indonesia · Hybrid
Skills: Business Strategy · Management · Communication · Marketing
 Feez - Belajar Coding Dengan Harga Terjangkau Sekarang!

Chief Technology Officer
Yayasan Pelatihan Sains Pelajar Indonesia · Part-time
Jun 2019 - Present · 4 yrs 11 mos
Bekasi, West Java, Indonesia
Skills: JavaScript · PHP · Ajax · JQUERY · MySQL
 forpelind · forpelind
forpelindo - Penyelenggara Kompetisi Sains Online. Berkarya Untuk Negeri , Menciptakan Kemajuan Untuk Pendidikan IndonesiaForum Pelatihan Sains Pelajar Indonesia merupakan...

Website Developer
PT ANTARES DWIMA PRATAMA · Part-time
Jan 2019 - Present · 5 yrs 4 mos
 PT ANTARES DWIMA PRATAMA
PT. Antares Dwima Pratama was established in 2014 by Notary Public's memorandum of Association. The main activities of the company are in Marine Surveying & Consultancy, Carg...

List of Contents

- *Pengenalan Konsep Pemrograman*
- *Syntax Dasar C++ : Variabel & Tipe Data, Masukan & Keluaran, Percabangan (if-else), Perulangan for – while*
- *Pengenalan Algoritma*
- *Problem Solving Soal Pemrograman Dasar*

Disclaimer

- *Kalau aku kecepetan bilang*
- *Kalau **koneksiku putus – putus, pinjam dulu seratus***
- *Kalau ada typo codingan bilang ya ges ya, aku udah kelamaan dihantam Python soalnya balik ke C++ kadang suka lupa titik koma atau tipe data ☺*

01

Konsep Pemrograman

Pada sub-materi ini anda diminta untuk memaksimalkan imajinasi anda, karena kali ini kita akan belajar dengan cara yang sedikit berbeda, materi akan dihubungkan kepada studi kasus di dunia nyata dan semoga anda lebih mudah memahaminya

Kenapa harus belajar Pemrograman?

Ingat gengs : “Kalian anteq OSN Informatika”



27. Perhatikan potongan program berikut!

```
if (a > b){  
    if (a > c) {  
        if (d > a) {  
            x = d * d;  
        } else {  
            x = a * a;  
        }  
    } else {  
        if (d > c) {  
            x = d * d;  
        } else {  
            x = c * c;  
        }  
    }  
} else {  
    if (b > c) {  
        if (d > b) {  
            x = d * d;  
        } else {  
            x = b * b;  
        }  
    } else {  
        if (d > c) {  
            x = d * d;  
        } else {  
            x = c * c;  
        }  
    }  
}
```

Jika nilai $a=12$, $b=23$, $c=45$, dan $d=78$, berapakah nilai dari x seprogram tersebut dijalankan?

Jawaban: { tuliskan jawaban dalam bentuk ANGKA saja }

MEMBUAT PROGRAM

3. Buatlah program menggunakan bahasa C/C++ sesuai deskripsi cerita di atas untuk menentukan total bobot kentang terkecil yang perlu dipindahkan oleh Pak Dengklek agar tujuan tercapai sesuai dengan peraturan di atas.

Format Masukan:
Baris pertama berisi dua buah bilangan: N (banyaknya kentang di truk A) dan M (banyaknya kentang di truk B). Baris kedua berisi N buah bilangan, menyatakan bobot-bobot kentang di truk A, sedangkan baris ketiga berisi M buah bilangan, menyatakan bobot-bobot kentang di truk B.

Format Keluaran:

Bidang Informatika/Komputer Halaman 3 dari 12 OSN-P 2023

Contoh Masukan dan Keluaran:

Contoh Masukan	Contoh Keluaran
3 4 5 1 3 8 5 4 9	17
1 4 2 1 5 5 5	15

Pertahanan Bogor

Indonesian (id) Switch

Time limit 2 s
Memory limit 512 MB

Deskripsi
Bogor sedang diserang monster jahat. Pak Dengklek sedang berada di kapal pertahanan yang terletak di sungai Cisadane. Sungai Cisadane dibagi menjadi 10^6 bagian yang dinomori dari 1 sampai 10^6 . Pada awal hari ke- i , kapal Pak Dengklek berada pada bagian ke- i dari sungai. Pada akhir hari setiap hari, Pak Dengklek harus melakukan salah satu dari hal berikut tepat satu kali:

- Mengikuti arus sungai untuk menggerakkan kapalnya dari bagian ke- i ke bagian ke- $(i+1)$ dari sungai, dengan z melambangkan nomor bagian tempat kapal berada saat ini.
- Menggunakan teknologi ajib untuk membuat kapalnya langsung berada di bagian ke-1 dari sungai.

Monster akan menerjang sungai selama N hari dari hari ke-1 sampai hari ke- N . Pada awal dari masing-masing hari, monster akan menerjang seluruh bagian sungai kecuali satu bagian. Formalnya, untuk setiap $1 \leq i \leq N$, bagian ke- A_i dari sungai adalah zona aman pada hari ke- i . Ini berarti, kapal Pak Dengklek tidak akan terkena serangan pada hari ke- i jika dan hanya jika kapalnya berada pada bagian ke- A_i sungai pada awal hari itu.
Banyaknya minimum serangan yang mengenai kapal Pak Dengklek jika Pak Dengklek menggerakkan kapalnya secara optimal?

Batasan

- $1 \leq N \leq 200\,000$
- $1 \leq A_i \leq N$

Subsoal

- (5 point) Hanya berisi kasus uji berikut:
 $N = 10$
 $A = [2, 1, 2, 3, 2, 3, 4, 3, 4, 5]$
- (3 point) $N \leq 8$
- (7 point) $N \leq 17$
- (7 point) $A_1 = A_2 = \dots = A_N$
- (11 point) $A_i \leq 2$
- (21 point) $N \leq 200$
- (11 point) $N \leq 3000$
- (10 point) $A_i \leq 200$
- (20 point) Tidak ada batasan tambahan.

Masukan

Yaudah Kita
Pelajarin
aja ya Pelan Pelan

Apa itu Pemrograman?

Coba kamu bayangkan ada Seorang Ibu yang memberikan tugas tertulis di pintu kulkas kepada anaknya Bernama Ani seperti di bawah ini :

Buat Sweetie Pookie Bear Sayangnya Mamah, Ani :

Mamah liburan bentar ya ke Jogja kamu jaga Rumah.
Setiap 2 Minggu sekali Rumah disapu, Cuci Pakaian
deterjennya 2 liter / 1kg baju ya

Anak Lagi
Nganggur gaada
kerjaan.

Anak Lagi Push
Emel / Ngerjain
Sesuatu



Aku Diam

Waktunya Disuruh
Suruh

Apa itu Pemrograman?

Dari Contoh di atas kita dapat menilai bahwa Ibunya Ani :

- Memberikan perintah kepada Ani untuk melakukan suatu pekerjaan
- Pekerjaan yang dilakukan dapat berulang kali

Setiap 2 Minggu sekali Rumah disapu,



Apa itu Pemrograman?

Dari kasus di atas yang dilakukan Ibu kepada Ani disebut sebagai pemrograman.

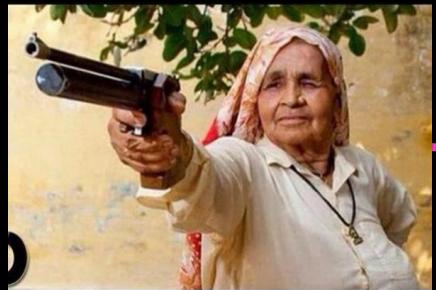
Pemrograman adalah proses menulis suatu program yang terdiri dari sekumpulan instruksi atau perintah untuk melakukan pekerjaan tertentu

```
procedure lakukan_pekerjaan:  
Sapu_rumah(2x sehari);  
cuci_pakaian({  
detergen = 2 liter x berat baju  
})
```

Apa itu Pemrograman?

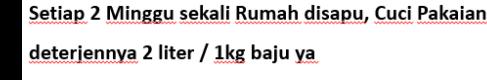
Dari kasus di atas yang dilakukan Ibu kepada Ani disebut sebagai **pemrograman**.

Hanya saja dalam konteks Teknologi Ibu itu adalah kita / programmer (orang yang menulis program) lalu Ani adalah Komputer.



Apa itu Pemrograman?

Dalam melakukan pekerjaannya Ani pasti bisa melaksanakan pekerjaannya dengan baik jika

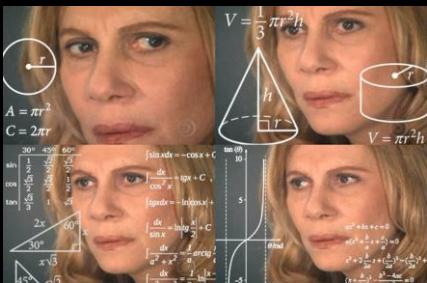
- Ada parameter pekerjaan yang jelas -> 
Setiap 2 Minggu sekali Rumah disapu, Cuci Pakaian
deterjennya 2 liter / 1kg baju ya
- Sehingga pemrograman bisa disebut efektif jika ada parameter yang jelas

Bagaimana Cara Memprogram?

Jika dilihat dari yang dilakukan ibu tadi :

Ibu Menulis Perintah (**Coding**)

Proses menulis ini kita sebut sebagai
Coding

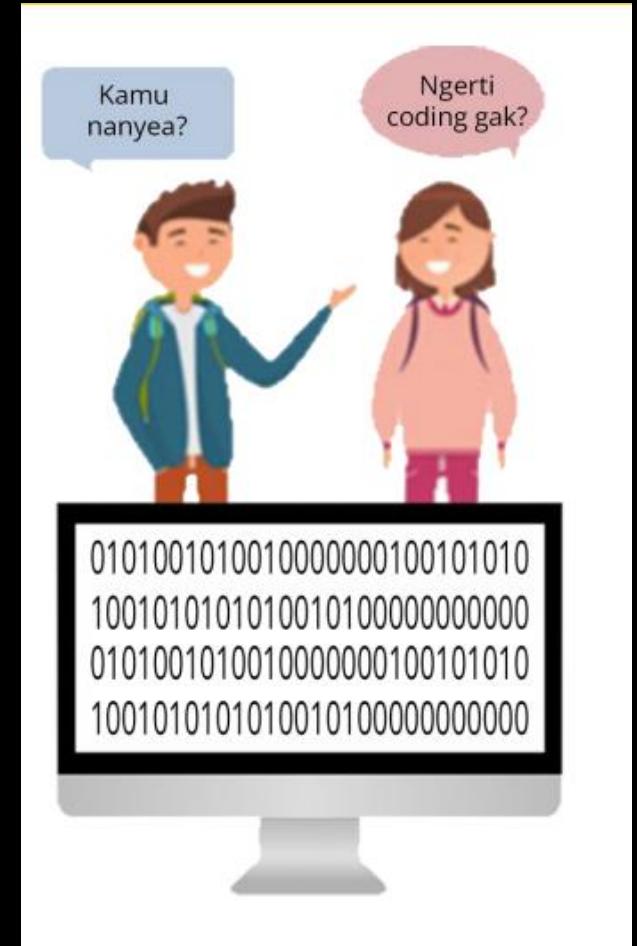


Si Ani membaca dan memahami
Perintah ibu (**Compiling**)

Proses memahami perintah oleh Ani
disebut sebagai **Compiling**

Coding

Pada dasarnya, komputer tidak mengerti Bahasa Manusia sehingga untuk memerintahkan computer melakukan sesuatu, Manusia harus menuliskan perintahnya dalam Bahasa Pemograman Komputer. Proses penulisan itulah yang disebut Coding.



Coding Vs Programming

Menjadi topik yang lumayan hangat dibicarakan.

Coding dan Programming sebenarnya adalah dua hal yang tidak sama

Karena sebenarnya Coding adalah sebagian dari proses Programming dan ini berarti Coding belum tentu programming

Coding Vs Programming

Kumpulan perintah bisa disebut sebagai program dan menjadi bagian dari Programming jika :

- Jika di dalamnya terdapat parameter pengukuran yang jelas
- Perintah bisa dilaksanakan berulang kali (dinamis)
- Perintah biasanya diintegrasikan dengan Formula, Rumus, atau Permodelan Matematis / Logika.

Bagaimana Cara Memprogram?

Untuk memprogram kita membutuhkan :

- **Dalam Proses Coding**
 1. Kita membutuhkan Bahasa yang sesuai dengan kebutuhan program kita.
 2. Kode ditulis dalam format text/script dan membutuhkan media menulis kode dan ini disebut sebagai **[IDE*]: Text Editor, Tools, Environment**

Bagaimana Cara Memprogram?

Untuk memprogram kita membutuhkan :

- **Dalam Proses Compiling**

1. Kita membutuhkan Compiler. Yaitu suatu mesin yang membaca perintah yang kita tulis lalu mengeksekusinya. Juga akan ditemui *Interpreter (Pemroses Lain)* tapi sejatinya bekerja dengan fungsi yang sama. Ibaratkan otak pada manusia.

Bahasa Pemrograman

Bahasa adalah standar format perintah yang kita berikan. Seperti manusia dalam Pemrograman Bahasa Juga ada banyak. Variasi Bahasa Pemrograman dikarenakan setiap jenis program memiliki kebutuhan yang berbeda – beda dalam penulisan perintah.

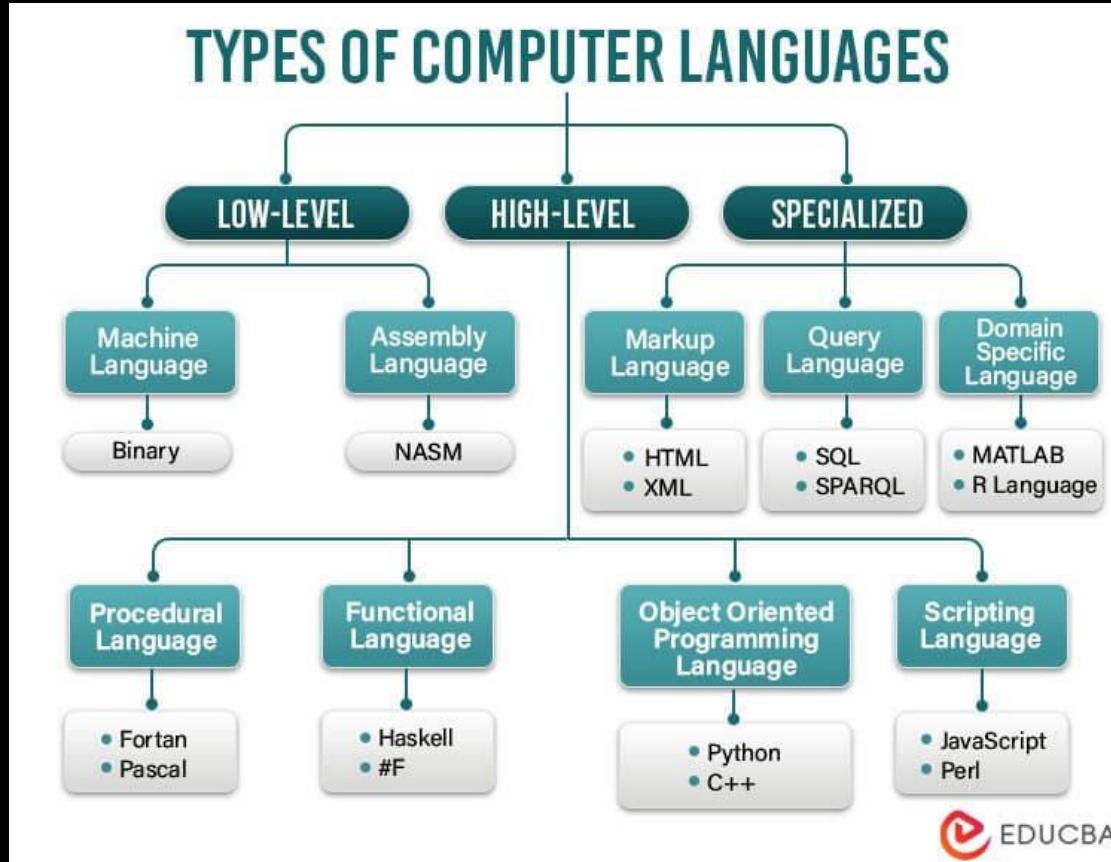
Inggris	I don't care
Indonesian	Bodo amat
Jawa	Luweh

Bahasa Pemrograman

Walaupun variasi Bahasa yang banyak. Tapi sejatinya setiap Bahasa Pemrograman selalu memiliki tujuan perintah yang serupa. Terkadang yang menjadi pembeda setiap Bahasa Pemrograman adalah :

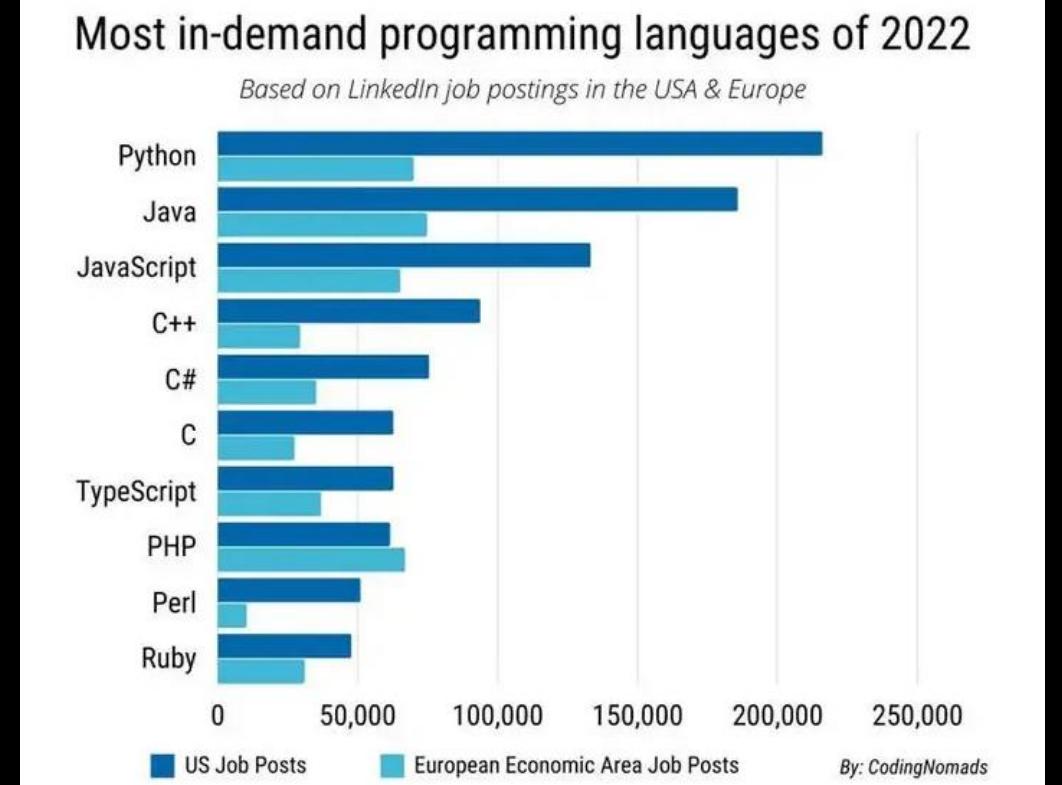
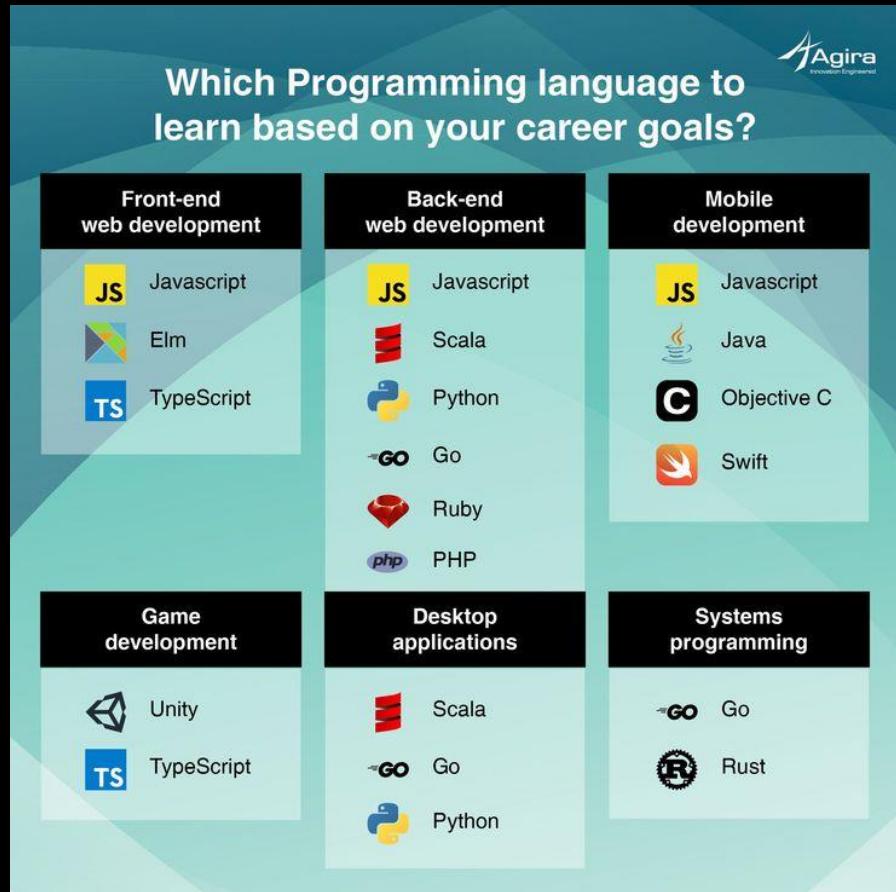
- Kemudahan penulisan
- Kecepatan Waktu Proses
- Alokasi Memori Komputer

Bahasa Pemrograman

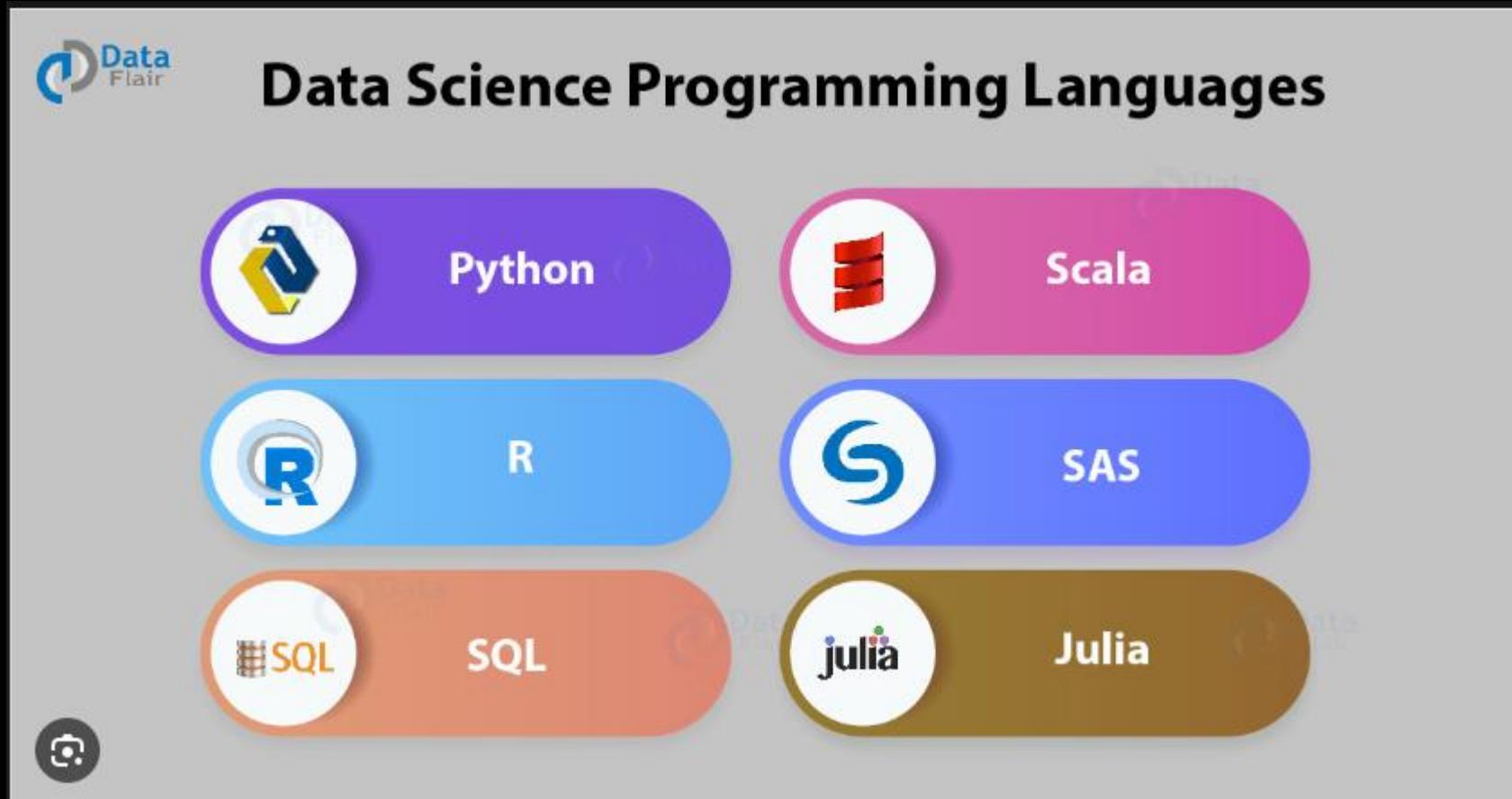


Semakin tinggi level bahasanya, semakin mendekati Bahasa manusia dan mudah dipahami.

Bahasa Pemrograman



Bahasa Pemrograman



Bahasa Pemrograman



Bahasa Pemrograman

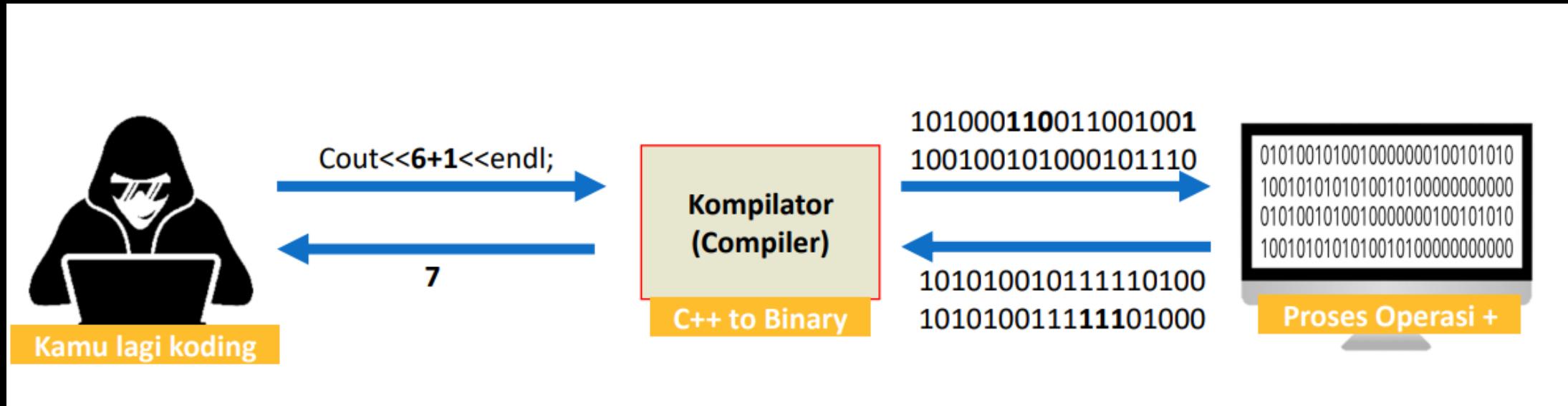
Untuk OSN Informatika sendiri Format Bahasa Pemrograman yang digunakan untuk mengerjakan soal adalah C++.

FYI : Umumnya C++ banyak digunakan dalam pembelajaran Programming karean banyak Bahasa Pemrograman lain yang sebenarnya adalah turunan dari Bahasa Pemrograman C++

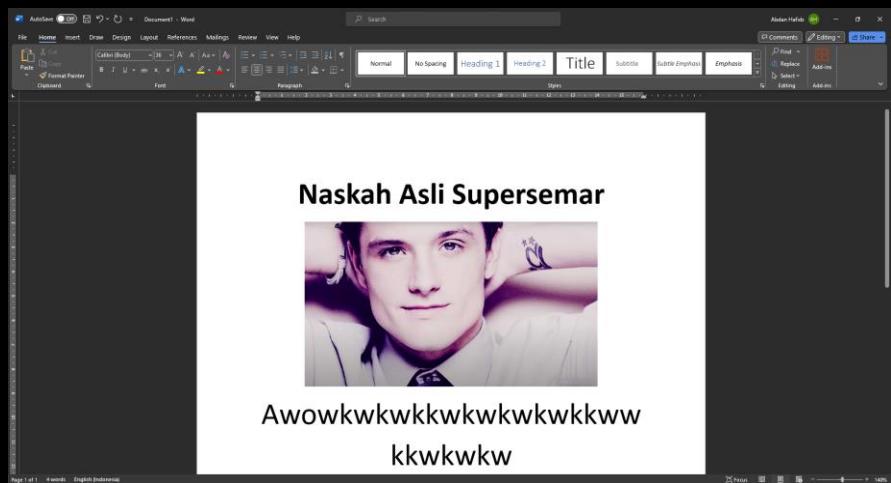
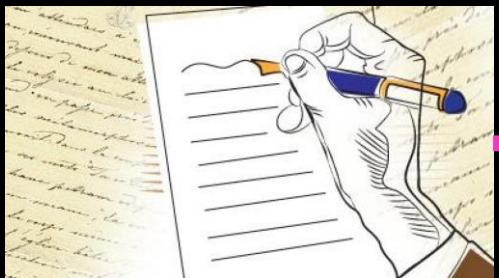
Compiler

- Syntax-syntax tiap Bahasa pemograman telah ditetapkan, sehingga akan memudahkan Kompilator mengerti kode yang kita tulis.
- Nantinya, Bahasa pemograman yang telah kita tulis akan diterjemahkan kedalam sebuah Kompilator supaya dapat dimengerti dan dieksekusi oleh Komputer.
- Compiler = program komputer yang dapat menerjemahkan bahasa pemrograman tingkat tinggi ke bahasa mesin

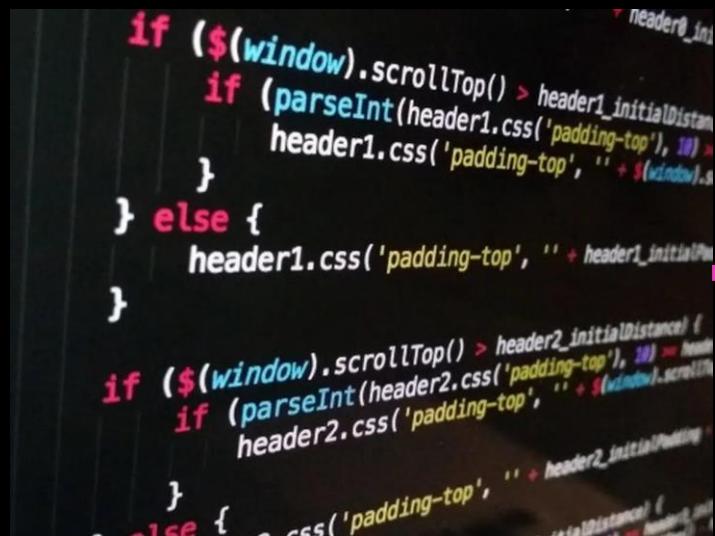
Compiler



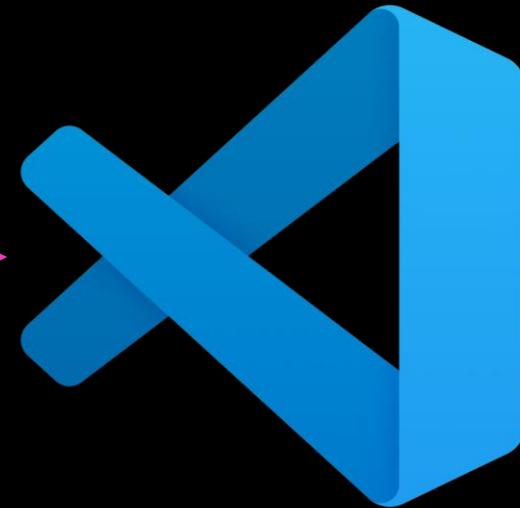
IDE (Integrated Development Env.)



IDE (Integrated Development Env.)



```
if ($window.scrollTop() > header1_initialDistance) {  
    if (parseInt(header1.css('padding-top'), 10) <= header1.css('padding-top', '' + $window.scrollTop())) {  
        header1.css('padding-top', '' + $window.scrollTop());  
    }  
}  
else {  
    header1.css('padding-top', '' + header1_initialPadding);  
}  
  
if ($window.scrollTop() > header2_initialDistance) {  
    if (parseInt(header2.css('padding-top'), 10) <= header2.css('padding-top', '' + $window.scrollTop())) {  
        header2.css('padding-top', '' + $window.scrollTop());  
    }  
}  
else {  
    header2.css('padding-top', '' + header2_initialPadding);  
}
```



02

Mulailah Memprogram

Pada sub-materi ini diharapkan anda memperhatikan seluruh tutorial yang diberikan. Pembelajaran bersifat runut dan setiap materi saling berkaitan satu sama lain. Jika anda tertinggal Sebagian materi maka anda akan sulit lanjut ke materi berikutnya.

***JIKA ADA HAL YANG DIBINGUNGKAN MOHON
LANGSUNG DITANYAKAN!***



Perhatikan Lalu Praktikkan!

*Gak boleh gak focus, gak boleh gak dipraktikkan, gak boleh gak
paham*

Persiapan dan Instalasi

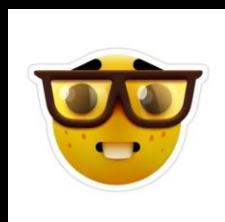
1. IDE/Compiler/Text Editor



With VSCode :

<https://code.visualstudio.com/docs/languages/cpp>

(Agak Ribet)



With DevCPP :

<https://sourceforge.net/projects/orwelldevcp>

(Lumayan Mudah)



With OnlineGDB :

https://www.onlinegdb.com/online_c++_compiler

(Turu Deck)

Lanjut

AYO NGODINGG!!!

2.1

Struktur C++

Header Files , Namespace, Main Function

Struktur Dasar C++

Fungsi Main

```
int main() {  
cout<<"Hello World";  
return 0;  
}
```

Struktur Dasar C++

1. **int main()** : Deklarasi fungsi main, semua sintaks/perintah yang ditulis di dalam fungsi main yang akan dijalankan oleh compiler.
2. **cout<<“Hello World”** : Perintah untuk menampilkan output/keluaran pada layar adalah cout<<“isi kalimat”.
3. **return 0;** : Karena fungsi main bertipekan integer (Bilangan Bulat ...,-3,-2,-1,0,1,2,3, ...) maka kita memberi keterangan bahwa jika program selesai nilai kembalinya adalah 0 (B aja). Nanti akan dibahas lebih lanjut di submateri berikutnya

Struktur Dasar C++

Header Files

```
#include<iostream> → Header File  
int main() {  
cout<<"Hello World";  
return 0;  
}
```

Tanpa
mencantumkan
Header File
`#include<iostream>`

Maka sintaks
cout,cin dll tidak
bisa digunakan

Struktur Dasar C++

Header Files

Header Files diibaratkan sebagai menu yang terdiri dari Kumpulan perintah – perintah yang sudah disusun dan siap kita pakai, sehingga lebih mudah dan cepat dalam menyusun program. Deklarasi header file adalah :

```
#include <namafileheader>
```

Struktur Dasar C++

Header Files

Tidak dengan Header Files

```
goreng_ayam() {  
panasin_minyak(suhu=80);  
cuci_ayam();  
masukkan_ke_wajan();  
}
```



Struktur Dasar C++

Header Files

Dengan Header Files

```
#include <wartegbahari>
pesen_ayam_goreng();
```



Struktur Dasar C++

Header Files

- **File Header** ada banyak macam, semuanya memiliki fungsi yang berbeda-beda.

Nama File Header	Fungsi yang disediakan File Header
iostream.h	<code>cin</code> : masukkan input <code>cout</code> : tampilkan output <code>endl</code> : kelang satu baris
stdio.h	<code>printf()</code> : tampilkan output <code>scanf()</code> : masukkan input
math.h	<code>sin()</code> , <code>cos()</code> , <code>tan()</code> : menghitung nilai sinus, cosinus, dan tangen dari suatu sudut
bits/stdc++.h	Hampir semua fungsi yang disediakan oleh File Header lainnya
Lainnya	Bisa dilihat via link berikut https://www.petanikode.com/cpp-sintaks/

Struktur Dasar C++

Namespace

```
#include<iostream> → Header File  
using namespace std; → Namespace  
int main() {  
cout<<"Hello World";  
return 0;  
}
```

Walaupun sudah menggunakan header file ‘iostream’ cout masih tidak berjalan karena sebenarnya untuk memanggil cout harus menggunakan sintaks

std::cout<<

Struktur Dasar C++

Namespace

```
#include<iostream> → Header File  
using namespace std; → Namespace  
int main() {  
cout<<"Hello World";  
return 0;  
}
```

Nah, agar kita tidak perlu repot – repot menulis std:: karena itu melelahkan kita bisa gunakan **using namespace std;**

Struktur Dasar C++

Header Files

Tidak dengan Namespace

```
#include<iostream>
using namespace std;
int main() {
    std::cout<<"Hello World";
    return 0;
}
```

Struktur Dasar C++

Namespace

Dengan Namespace

```
#include<iostream>
using namespace std;
int main() {
cout<<"Hello World";
return 0;
}
```

2.2

Variabel & Tipe Data

Variable, Tipe Data Dasar (Integer (int,longint) , String, Character, Boolean, Double, Float)

Variabel

Variabel adalah istilah dalam dunia matematika, yang memetakan sebuah nama ke suatu nilai

```
namadepan = "Abdan";  
namabelakang = "Hafidz";  
Jumlahpacar = 0;  
namalengkap = namadepan + namabelakang;
```

Variabel

Aturan penamaan variable :

1. Terdiri dari kombinasi karakter huruf, angka, dan underscore (_).
2. Tidak boleh dimulai dengan angka.
3. Huruf kapital dan huruf kecil dianggap berbeda. Artinya "a1" dan "A1" dianggap merupakan dua variabel yang berbeda.
4. Tidak boleh merupakan reserved word. Contoh reserved word pada C++: int, if, while, for, atau switch.



Tipe Data

Tipe Data Manusia :

Data	Tipe Data Bagi Manusia
1, 2, 69, 30001288, 012736, 2.56, 3.14	Angka
Kucing, anjing, rubah	Huruf, kata
a, b, c, d, _, :, @	Huruf, karakter
Andi sedang belajar., Kamu lagi apa?,	Kalimat

Tipe Data

Tipe Data Dalam C ++ :

Type	Definition	Control Character	Limits
int	Integer		-2147483648 to 2147483647
short	Short Integer		-32768 to 32767
long	Long Integer	l or L	-2147483648 to 2147483647
float	Floating Decimal Number	f or F	1.17549e-038 to 3.40282e+038
double	Double Decimal Number		2.22507e-308 to 1.79769e+308
long double	Long Decimal Number		2.22507e-308 to 1.79769e+308
char	Character		-128 to 127
unsigned int	Unsigned Integer		0 to 4294967295
unsigned short	Unsigned Short Integer		0 to 65535
unsigned long	Unsigned Long Integer		0 to 4294967295
unsigned char	Unsigned Character		0 to 255
bool	True or False		True = 1 and False = 0

Yang paling sering dipakai :

- Integer : int,long long
- Decimal : double,float
- Kalimat & karakter : string, char
- Proposisi (Boolean) : bool

Tipe Data

Tipe Data Dalam C ++ :

Data	Tipe Data di C++
1, 2, 69, 30001288, 12736	<ul style="list-style-type: none">• int (untuk bilangan bulat yang kecil)• long long (untuk bilangan bulat yang lebih besar)
0.256, 7.21, 3.141592653589	<ul style="list-style-type: none">• Float• Double = Menampung angka lebih banyak di belakang koma
Kucing, anjing, rubah, a, b, _, @, Andi sedang belajar	<ul style="list-style-type: none">• string
a, b, c, d, _, :, @	<ul style="list-style-type: none">• char

2.3

Assignment, Masukan dan Keluaran

Variable Declaration, Assignment, cin (Input), cout (Output)

Deklarasi Variabel

Variabel adalah istilah dalam dunia matematika, yang memetakan sebuah nama ke suatu nilai

```
<tipedata> <namavariabel>
int x;
string kalimat;
long long x,y;
```

Assignment

Assignment artinya memberikan nilai pada variabel

```
<tipedata> <namavariabel> = <nilai>
int x = 9; → x bernilai 9 (x=9)
int y = 5; → x bernilai 5 (x=5)
int jumlah = x + y, kurang = x - y;
```

Variabel jumlah akan bernilai hasil $x + y = 9 + 5 = 14$ dan kurang = $9 - 5 = 4$

Assignment

Assignment artinya memberikan nilai pada variabel

```
string namadepan = "Abdan";  
string namabelakang = "Hafidz";  
string namalengkap = namadepan + " " + namabelakang;  
namalengkap = "Abdan Hafidz"                                spasi
```

Assignment

Assignment artinya memberikan nilai pada variabel

```
int jumlahpacar = 0;  
bool punyapacar = jumlahpacar > 0;  
bool jomblo = true;
```

*Karena jumlah pacar tidak >0
Maka variable punyapacar akan
bernilai false.*

Ini dinamakan sebagai proposisi

Assignment

Assignment artinya memberikan nilai pada variabel

```
double A = 5.0, B = 2.0;
```

```
double X = A/B;           X = 2.5 , karena tipe data X adalah double (Desimal)
```

```
int C = 5, D = 2;
```

```
int Y = C/D;             Y = 2 , karena X integer maka nilai dipaksa bulat dengan melakukan pembulatan ke bawah (floor) 2.5 -> 2 , div
```

Keluaran

Hasil dari assignment dapat ditampilkan di layar menggunakan cout<<

```
string namadepan = "Abdan";
string namabelakang = "Hafidz";
string namalengkap = namadepan + " " + namabelakang;
cout<<"Halo "<<namalengkap;
```

Output : Halo Abdan Hafidz

Keluaran

Penggabungan Output mendatar

```
string nama = "Abdan Hafidz";
int usia = 18;
char gender = "L"
cout<<"Halo :"<<nama<<" Usia :"<<usia;
```

Output : Halo :Abdan Hafidz Usia :18

Keluaran

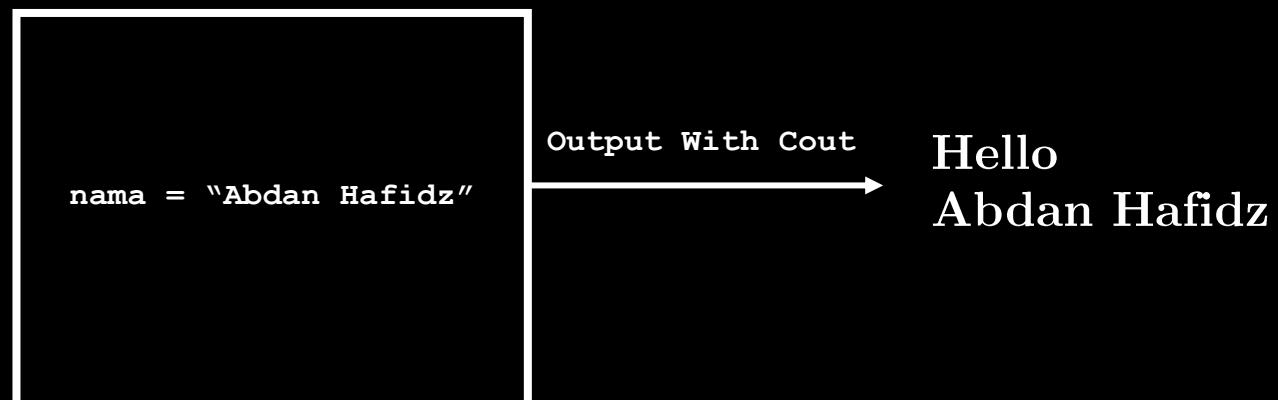
Penggabungan Output perbaris, gunakan endl untuk membuat baris baru

```
string nama = "Abdan Hafidz";
int usia = 18;
cout<<"Halo :"<<nama<<endl;
cout<<" Usia :"<<usia;
Output : Halo :Abdan Hafidz
Usia :18
```

Masukan

Kita ingin program kita menjadi lebih interaktif. Orang dapat menggunakan program kita dengan fleksibel.

Jika sebelumnya kita membuat program mencetak nama:

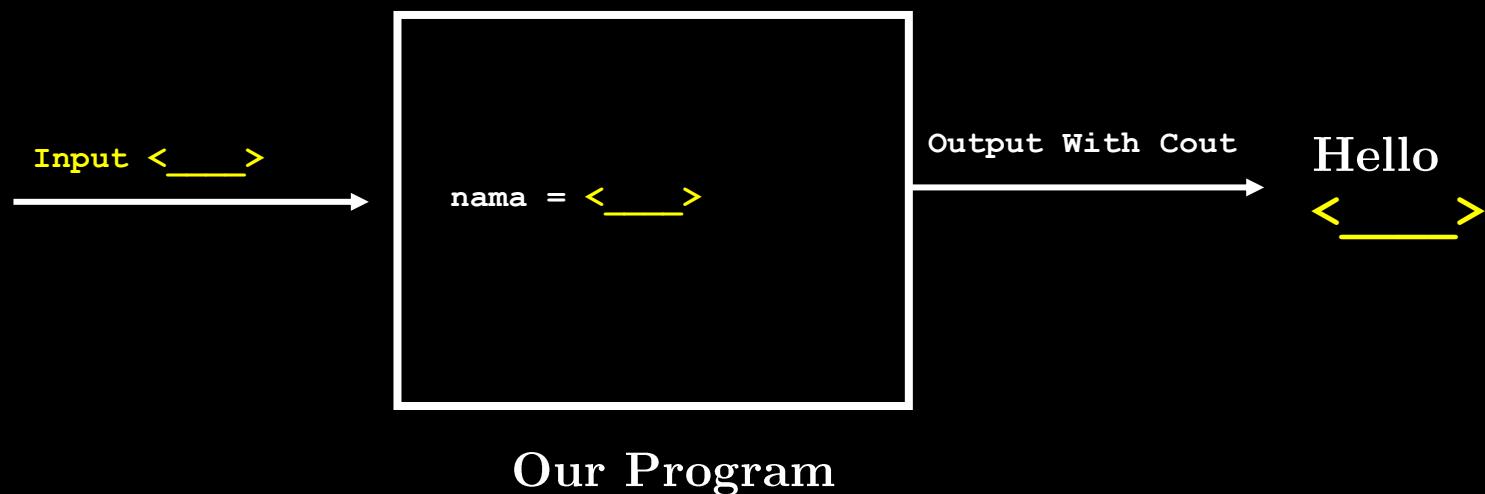


Our Program

Masukan

Kita ingin program kita menjadi lebih interaktif. Orang dapat menggunakan program kita dengan fleksibel.

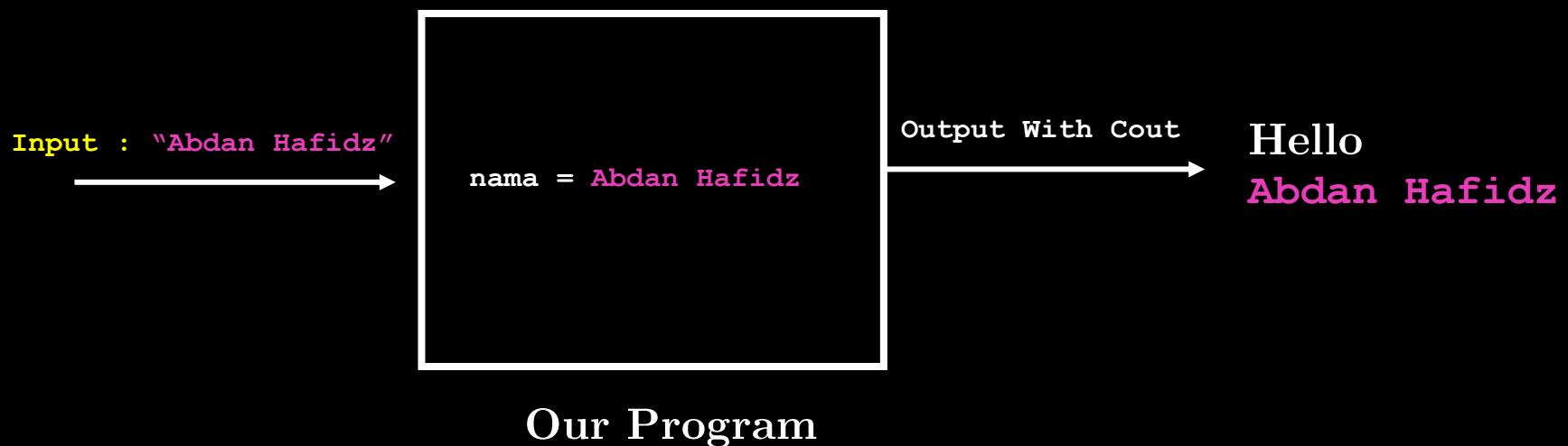
Jika sebelumnya kita membuat program mencetak nama:



Masukan

Kita ingin program kita menjadi lebih interaktif. Orang dapat menggunakan program kita dengan fleksibel.

Jika sebelumnya kita membuat program mencetak nama:



Masukan

Untuk menggunakan input, kita bisa memanfaatkan
cin>>

```
string nama;  
cin>>nama;  
cout<<"Halo :"<<nama<<endl;
```

Operator C++

$$\begin{aligned} & 3 / 2 \\ & 3 \bmod 2 \\ & \frac{1}{2\sqrt{3}} \\ & \frac{1}{2} - \\ & \textcircled{1} \\ & 3 - 2 = 1 \\ & 3 \bmod 2 = 1 \end{aligned}$$

Operators in C++

Operator	Type
$++$, $--$	Unary operator
$+$, $-$, $*$, $/$, $\%$	Arithmetic operator
$<$, \leq , $>$, \geq , $=$, $!=$	Relational operator
$\&\&$, $\ $, $!$	Logical operator
$\&$, $ $, $<<$, $>>$, \sim , \wedge	Bitwise operator
$=$, $+=$, $-=$, $*=$, $/=$, $\%=$	Assignment operator
$:$	Ternary or conditional operator

Increament - Decreament

Kita bisa menambahkan nilai dari variable yang sudah terassign :

```
int x = 5;  
x += 2; x sekarang adalah 7 karena x += 2 itu artinya x' = x + 2  
x ++; x sekarang adalah 8 karena x ++ itu artinya x' = x + 1  
x --; x sekarang adalah 7 karena x -- itu artinya x' = x - 1  
x -= 2; x sekarang adalah 5 karena x -= 2 itu artinya x' = x - 2
```

Increament - Decreament

Kita bisa menambahkan nilai dari variable yang sudah terassign :

```
string nama = "Abdan"  
nama += " Hafidz"; nama sekarang adalah "Abdan Hafidz"  
karena nama += " Hafidz" itu artinya  
nama' = nama + " Hafidz"
```

Latihan Dulu Brow

Deskripsi Soal

Sebuah Apel dihargai sebesar N dan sebuah Jeruk dihargai sebesar M . Jika Andi membeli X buah Apel dan Y buah Jeruk. Tentukanlah berapa harga yang harus dibayar Andi!

Format Input

Satu baris berisi bilangan $N \ M \ X \ Y$

Format Output

Sebuah baris berisi jawaban harga yang harus dibayar Andi.

2.4

Percabangan

if-else

Percabangan

Percabangan merupakan ekspresi program yang akan memvalidasi suatu kondisi. Jika suatu kondisi terjadi (bernilai benar) maka perintah terkait akan dilaksanakan, dan jika kondisi itu terjadi (bernilai salah) memungkinkan program melakukan perintah lainnya.

Perfect if else statement



if-else

Jika maka. ..., jika tidak maka....

```
if (<kondisi>){  
    <perintah dijalankan jika  
<kondisi> benar >  
} else{  
    <perintah dijalankan jika  
<kondisi> salah >  
}
```

if-else

Jika maka. ..., jika tidak maka....

```
int x = 9;  
if(x == 9) {  
    cout<<"X sama dengan Sembilan";  
} else{  
    cout<<"X tidak sama dengan Sembilan";  
}
```

if-elseif

```
if(<kondisi1> {  
    <perintah dijalankan jika  
<kondisi1> benar >  
} else if(<kondisi2> {  
    <perintah dijalankan jika  
<kondisi1> salah dan  
<kondisi2> benar >  
}
```

if-elseif

Jika maka. ..., jika tidak maka....

```
int x = 9;
if (x >= 9) {
    cout<<"X lebih dari sama dengan 9";
else if (x>6) {
    cout<<"x memenuhi 6<x<9";
} else{
    cout<<"x ≤ 6";
}
```

Nested if

```
if(<kondisi1> {
    if(<kondisi2> {
        <perintah dijalankan jika kondisi 1 dan 2 terpenuhi>
    }else{
        <perintah dijalankan jika kondisi 1 terpenuhi tapi 2 tidak>
    }
}else if(<kondisi3> {
    if(<kondisi4> {
        ...
    }else{
        ...
    }
}
```

Clean Logical Nested If

Logic Operator

```
if (<kondisi1> && <kondisi2>) {  
    <perintah dijalankan jika kondisi  
    1 dan 2 terpenuhi>  
} else {  
    <perintah dijalankan jika kondisi  
    1 atau 2 tidak terpenuhi>  
}
```

Logic Operator

Logical Operators

Operator	Meaning	Example	Result
<code>&&</code>	Logical and	<code>(5<2)&&(5>3)</code>	False
<code> </code>	Logical or	<code>(5<2) (5>3)</code>	True
<code>!</code>	Logical not	<code>!(5<2)</code>	True

Latihan Dulu Brow

Deskripsi Soal

Buatlah sebuah formulir biodata yang berisikan data :

Nama Depan, Nama Belakang, Nama Lengkap, Usia, Gender (L/P), Sudah Menikah / Belum (1 = Sudah, 0 = Belum).

Contoh Input

Nama Depan:Abdan

Nama Belakang:Hafidz

Usia:18

Gender:L

Sudah:0

Contoh Output

Seseorang Bernama Abdan Hafidz berusia 18 tahun adalah seorang laki – laki , dan belum menikah

2.5

Pengulangan

for, while, do-while

Pengulangan

Misalkan kita diminta untuk mencetak angka 1 – 5.
Mungkin kita bisa saja membuat program

```
cout<<1<<endl;  
cout<<2<<endl;  
cout<<3<<endl;  
cout<<4<<endl;  
cout<<5<<endl;
```

Pengulangan

Tapi bagaimana jika kita ingin mencetak angka 1 – 10 (oke sih gak terlalu banyak), 1- 100? , 1-1000?, 1-10.000?, dan 1-n untuk n berdasarkan input yang diberikan? (Apa gak modar?)

Pengulangan

Untuk mengatasi ini kita bisa menggunakan perulangan for dan while.

for

For adalah jenis perulangan yang bisa digunakan untuk Batasan kondisi yang biasanya lebih spesifik dan tentu

```
for (<perintah_awal>; <kondisi>; <perubahan>) {  
    <kumpulan_perintah>  
}
```

for

Jika dari soal yang tadi kita ingin mencetak angka dari 1 – 10 misalnya.

```
for (int i = 1; i<=10; i++) {  
    cout<<i<<endl;  
}
```

$i = 1$
 $i = 2$
 $i = 3$
...
...

$i = 10$

Lebih Mudah
bukan??

for

For yang menurun

```
for (int i = 10; i>=1; i--) {  
    cout<<i<<endl;  
}
```

i = 10) -1
i = 9) -1
i = 8) -1
i = 7) 1
.....
i = 1

i = 1

for

Menampilkan bilangan ganjil

$$u_n \leq 10$$
$$u_{n \max} \rightarrow 9$$

```
a = u1   un max  
for (int i = 1; i <= 10; i += 2) {  
    cout << i << endl;  
}
```

bedanya

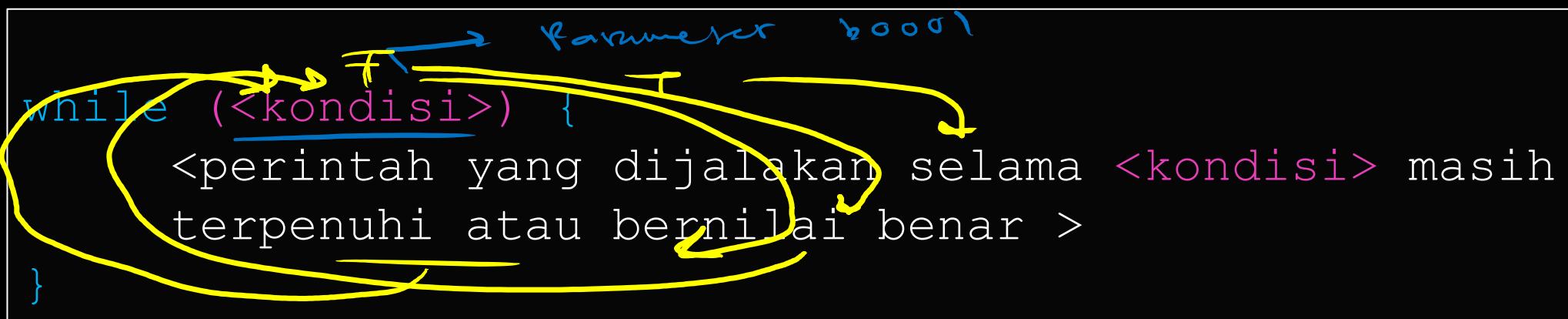
$$u_n$$
$$a = 1$$
$$\frac{b = 2}{n_{\max} = 5}$$
$$2n - 1 \leq 10$$
$$2n \leq 11$$
$$n \approx 5$$

$$u_n = a + (n-1)b$$
$$= 1 + (n-1)2$$
$$= 1 + 2n - 2$$
$$= 2n - 1 \rightarrow \text{bil ganjil}$$

$n_{\max} = \text{banyak Pemangaman / sisa}$
 $u_{n \max} = 2 \cdot 5 - 1 = 9$ \Rightarrow berhenti ketika $i = 9$

While

While memiliki fungsi yang sama seperti for yaitu melakukan pengulangan (iterasi) hanya saja berbeda dalam penulisan dan while bisa menangani kasus tertentu seperti perulangan tak tentu.



Pernyataan yang berulang

While

Kasus yang sama Jika dari soal yang tadi kita ingin mencetak angka dari 1 – 10 misalnya.

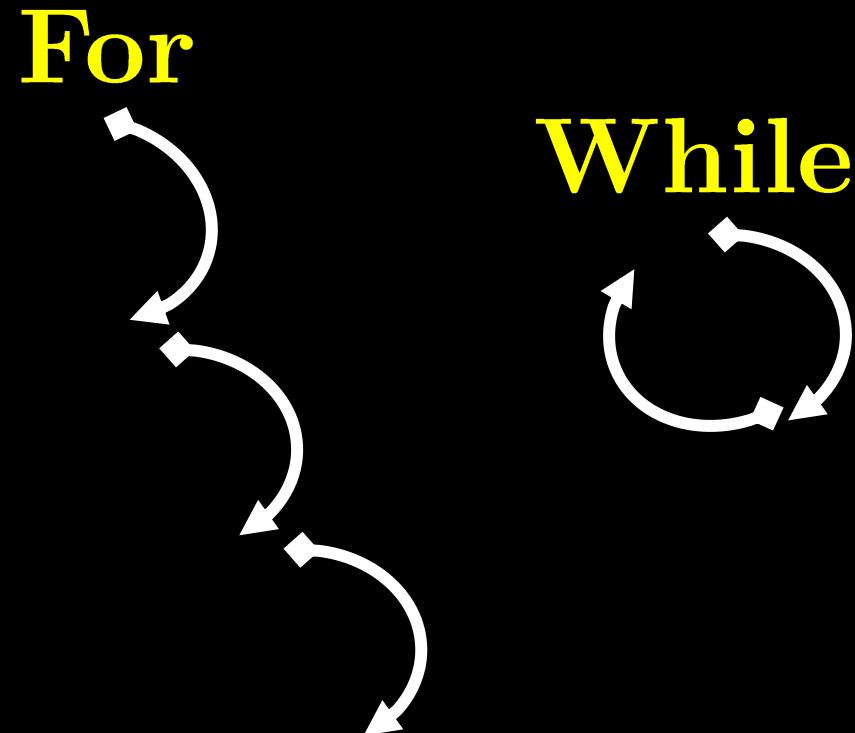
```
int i = 1;    )0
while(i<=10) {
    cout<<i<<endl;
    i++;
}
```

$i = 1 \rightarrow$ cetak 1
 $i = 2 \rightarrow$ cetak 2
 $i = 3 \rightarrow$ cetak 3
- - -
 $i = 10 \rightarrow$ cetak 10
~~i~~

For vs While

Secara umum for lebih rapih dibanding while. For lebih wajar untuk menentukan perulangan tentu

For VS While Loop	
Comparison Chart	
For Loop	While Loop
The for loop is used for definite loops when the number of iterations is known.	The while loop is used when the number of iterations is not known.
For loops can have their counter variables declared in the declaration itself.	There is no built-in loop control variable with a while loop.
This is preferable when we know exactly how many times the loop will be repeated.	The while loop will continue to run infinite number of times until the condition is met.
The loop iterates infinite number of times if the condition is not specified.	If the condition is not specified, it shows a compilation error.



While EOF

While EOF adalah iterasi yang bisa terus menerus sampai kiamat menerima input kecuali diterminasi (dihentikan paksa) dengan Ctrl + C jika di windows atau enter 2x

```
string S;
while(cin>>S) {
    cout<<S<<endl;
}
```

ga ngerainin

Parameter = False

Program tersebut akan terus menerus menerima input S lalu mengeluarkan output S

Do-While



Do-While

```
bensin = 9;  
while (bensin>0) {  
cout<<"Mobil jalan"<<endl;  
bensin--  
}
```

```
bensin=9;  
do {  
cout<<"Mobil jalan"<<endl;  
bensin--  
} while (bensin>0);
```

Do-While

```
bensin = 0;  
while (bensin>0) {  
cout<<"Mobil jalan"<<endl;  
bensin--  
}
```

Tidak ada Output

```
bensin=0;  
do {  
cout<<"Mobil jalan"<<endl;  
bensin--  
} while (bensin>0);
```

Tetap Mencetak “Mobil Jalan”

Nested Loop

Seperti if-else, for dan while juga bisa ditulis secara bersarang

```
for(int i = 0; i<5; i++) {  
    for(int j=1; j<i; j++) {  
        cout<<j<<endl;  
    }  
}
```

i = 0
while ~~j < 1~~ *j < 0*
i = 1 while (*kondisi2*) {
~~j = 1~~ while (*kondisi3*) {
i = 2 } *j > 1, j++*
~~j = 1~~ : *j < 2* ?
i = 3
j = 1 > j < 3
i = 2

Break

Break artinya menghentikan/interrupt pengulangan.

```
int sum = 1;  
while(sum++) {  
    if(sum == 10) {  
        break; Perulangan akan berhenti pada saat sum bernilai 10  
    }  
}
```

Latihan Dulu Brow

Deskripsi Soal

Pak Dengklek menemukan suatu pola. Untuk setiap bilangan bulat N tertentu akan dihasilkan output tertentu. Buatlah program yang sesuai dari contoh input-output di bawah ini

Contoh Input

3

Contoh Output

Contoh Input

4

Contoh Output

Latihan Dulu Brow

Deskripsi Soal

Pak Dengklek menemukan suatu pola. Untuk setiap bilangan bulat N tertentu akan dihasilkan output tertentu. Buatlah program yang sesuai dari contoh input-output di bawah ini

Contoh Input

3

Contoh Output

**

*

Contoh Input

4

Contoh Output

**

*

2.5

Tipe Data Array

Array / Larik

Masalah Array

Hampir serupa dengan permasalahan pengulangan.
Kita juga akan dikaitkan dengan efisiensi dari
kontinuitas program.

Masalah Array

Misalkan anda mempunyai 10 kotak, kotak ke satu sampai sepuluh berisikan bola berturut – turut sebanyak 1,0,2,3,4,4,5,6,7,0. Mungkin secara naif anda akan mendeklarasikan

```
int kotak_1 = 1;  
int kotak_2 = 0;  
int kotak_3 = 2;  
...  
Dst modar bro
```

Deklarasi Array

Untuk permasalahan ini kita bisa menggunakan Array atau Larik.

```
tipe data namavariabel[banyakdata] = {isi};
```

Deklarasi Array

Sehingga dari soal sebelumnya, karena jumlah kotak ada 10, maka banyak datanya ada 10. Dan bisa kita declare arraynya sebagai berikut :

```
int kotak[10] = {1,0,2,3,4,4,5,6,7,0};
```

Deklarasi Array

Sekarang anda diminta untuk mengeluarkan berapa banyak bola pada kotak pertama? Maka anda bisa menuliskan `kotak[0]` karena array memiliki indeks yang mulai dari nol.

```
int nilai_ke_n = namavariabel[n - 1];
```

```
int kotak [10] = {1,0,2,3,4,4,5,6,7,0};  
int isi_kotak_ke_1 = kotak[0];  
int isi_kotak_ke_2 = kotak[1];  
Dst...
```

Deklarasi Array

Sekarang anda diminta untuk mengeluarkan berapa banyak bola pada kotak pertama? Maka anda bisa menuliskan `kotak[0]` karena array memiliki indeks yang mulai dari nol.

```
int kotak [10] = {1,0,2,3,4,4,5,6,7,0};  
// tampilkan banyak bola pada kotak ke-1  
cout<<kotak[0]<<endl;
```

Deklarasi Array

Sekarang anda diminta untuk menampilkan berapa hasil jumlah isi bola pada kotak ke-5 dengan kotak ke-4

```
int kotak [10] = {1, 0, 2, 3, 4, 4, 5, 6, 7, 0};  
sum = kotak[4] + kotak[3];  
cout<<sum<<endl;
```

Deklarasi Array

Sekarang anda diminta untuk menampilkan berapa banyak isi bola pada masing – masing kotak. Anda bisa memanfaatkan perulangan for.

```
int kotak [10] = {1,0,2,3,4,4,5,6,7,0};  
for (int i = 0; i<10 ; i++) {  
    cout<<"Kotak ke-"<}
```

Deklarasi Array

Sekarang anda diminta untuk mengisi kotak yang kosong (bernilai 0) dengan 2 bola. Kita bisa memanfaatkan while/for. Agar variatif kali ini kita coba pakai while

```
int kotak [10] = {1,0,2,3,4,4,5,6,7,0};  
int index = 0;  
while(kotak[index]==0) {  
    if(index<10) {  
        kotak[index]+=2;  
        index++;  
    }else{  
        break;  
    }  
}
```

Deklarasi Array

Sekarang anda diminta untuk menghitung berapa hasil jumlah semua bola yang ada?

```
int kotak [10] = {1,0,2,3,4,4,5,6,7,0};  
int sum = 0;  
for (int i = 0; i<10 ; i++) {  
    sum+=kotak[i];  
}  
cout<<sum<<endl;
```

2.6

Sub-Program

Function, Procedure, Recursive Function.

Function

Adalah suatu pemetaan. Misalkan untuk x suatu bilangan rill. Kita bisa petakan ke sebuah fungsi $f(x) = x + 3$.

Yang berarti $f(1) = 4$, $f(2) = 5$, $f(3) = 6$, dst ...

Kita bisa menjadikan function sebagai sub program yang nantinya bisa dipanggil di program utama
int main()

Deklarasi Function

```
tipe data yang dikembalikan namafunction(<parameter>) {  
    return <operasi dengan <parameter>>  
}
```

Deklarasi Function

Contoh untuk $f(x) = x + 3$

```
int f(int x){  
    return x+3;  
}  
  
int main() {  
int y = f(3);  
    cout<<y<<endl;  
}
```

Contoh Penggunaan Function

Misalkan kita ingin menghitung berapa hasil kuadrat dari suatu bilangan x . Maka kita bisa gunakan $f(x) = x^2$

```
int f(int x){  
    return x*x;  
}  
  
int main(){  
    int x;  
    cin>>x;  
    int kuadrat_x = f(x);  
    cout<<kuadrat_x<<endl;  
}
```

Contoh Penggunaan Function

Misalkan kita ingin menghitung berapa luas segitiga dengan Panjang alas A, dan tinggi T.

```
double luas_segitiga(double A, double T) {
    return (A*T)/2;
}

int main() {
    int Alas, Tinggi;
    cin>>Alas>>Tinggi;
    cout<<"Luas :"<<luas_segitiga(Alas,Tinggi)<<endl;
}
```

Void Function

Void function kerap disebut sebagai procedure. Void artinya sebuah fungsi tidak mengembalikan nilai (tanpa return value) apapun, tapi bisa membuat tampilan.

Contoh seperti berikut ini :

```
void id_card(string nama, int usia, string waifu) {  
    cout<<"===== Kartu Member Anime ====="  
    cout<<"Nama : "<<nama<<endl;  
    cout<<"Usia : "<<usia<<endl;  
    cout<<"Waifu : "<<waifu<<endl;  
}
```

Recursive Function

Fungsi Rekursif adalah fungsi yang memanggil dirinya sendiri dalam operasi. Contoh sederhana adalah menentukan factorial. Silahkan perhatikan operasi matematika berikut.

Recursive Function [Factorial]

$$n! = n \times (n-1) \times (n-2) \times \dots \times 1$$

$$0! = 1$$

Misalkan $f(n) = n!$

$$f(n) = n \times (n-1) \times (n-2) \times \dots \times 1$$

$$f(n) = n \times (n-1)!$$

$$f(n-1) = (n-1)!$$

$$f(n) = n \times f(n-1), \text{ untuk } f(0) = 1$$

Recursive Function [Factorial]

Fungsi rekursif harus mempunyai kasus basis / awal.
Contoh untuk soal factorial adalah saat $f(0) = 1$.

$$\text{mutlak}(x) = \begin{cases} x, & x \geq 0 \\ -x, & x < 0 \end{cases}$$

Logic statement

Recursive Function [Factorial]

```

int f(int n) {
    if(n == 0) {
        return 1;  $f(n), n=0 \rightarrow 1$ 
    }  $f(0) = 1$ 
    else{
        return n*f(n-1);
    }
}

```

Base case

```

int main() {
    int n;
    cin>>n;
    int hasilFaktorial = f(n);
    cout<<n<<"! = "<<hasilFaktorial<<endl;
}

```

$$f(n) = \begin{cases} 1, & n=0 \\ n \times f(n-1), & n > 0 \end{cases}$$

$$f(n) = \underline{f(n-1)} + \underline{f(n-3)} \rightarrow f(1) \text{ s/d } f(3)$$

Recursive Function [Fibonacci]

Contoh lain Fungsi rekursif adalah Fibonacci.

Deret Fibonacci : 1,1,2,3,5,8, ...

$$\frac{f(1) \text{ dan } f(2)}{f(n) = f(n-1) + f(n-2)}$$

untuk $f(1) = 1$, $f(2) = 1$

Sebuah Fungsi Rekursif f
dgn orde - k
dia harus memiliki ^{minimal} base case

$$f(0) \text{ s/d } f(k-1)$$

$$\text{atau } f(1) \text{ s/d } f(2)$$

Orde Function

* Factorial \rightarrow orde 1

$$f(n) = n \times \underline{f(n-1)}$$

* Fibonacci \rightarrow orde 2

$$f(n) = f(n-1) + f(n-2)$$

Recursive Function [Fibonacci]

```
int f(int n) {
    if(n == 1 || n == 2) {
        return 1;
    } else{
        return f(n-1) + f(n-2);
    }
}

int main() {
    int n;
    cin>>n;
    int hasilFibonacci = f(n);
    cout<<"Fibonacci ke-"<
```

$$f(n) = \begin{cases} 1 & , 1 \leq n \leq 2 \\ f(n-1) + f(n-2), & > 2 \end{cases}$$

Permutasi n objek = $n!$
= Faktorialn

~~520~~ - 2 objek
↳ ~~20~~ objek

Kenapa Harus Function

Function atau subprogram biasa digunakan dengan tujuan tertentu, seperti :

- Kode yang kita tulis akan lebih nyaman dilihat dan lebih rapih.
- Memudahkan saat kolaborasi. Misal kita memprogram Bersama tim. Tim tidak perlu melakukan copy-paste Source Code. Anda hanya perlu mengarahkan untuk melakukan pemanggilan fungsi tertentu.
- Memory & Process Management. Dari sisi operating System akan ada yang Namanya Subroutine (**INI GAK PERLU DILANJUT GES GAK ADA DI OSN KOK**)

Legacy code → code kurang bagus
Clear code → cuma bisa dipakai
→ tidak boro)

03

Pengantar Algoritma

Pada Materi ini anda akan lebih banyak bertemu dengan konsep Langkah penyelesaian permasalahan.

Apa itu Algoritma?

Algoritma merupakan metode yang dirancang dalam menyusun instruksi – instruksi / perintah dalam program.

Algoritma juga bisa dikatakan sebagai konsep atau paradigma kita saat menyelesaikan permasalahan secara sistematis.

Apa itu Algoritma?

Contohnya, misalkan anda ingin memasak Mie Instan :



Langkah penyajian ini berisikan instruksi apa yang harus dilaksanakan untuk memasak mie instan.

Dan ini disebut sebagai Algoritma.

Apa itu Algoritma?

Algoritma juga menunjukkan bahwa metode penyelesaian terhadap suatu permasalahan itu sangat beragam, serta setiap metode memiliki kelemahan dan kelebihannya masing – masing.

Umumnya algoritma yang efektif memperhatikan Kompleksitas, Waktu Kompilasi Program, Memory Program, dan Penyelesaian yang runut serta sistematis.

Apa itu Algoritma?

THERE ARE TWO TYPES OF PEOPLE



Apa itu Algoritma?

“There are many ways to do 16×15 !”

$$16(10) + 16(5) = 160 + 80 = 240$$

$$16 \times 15 = 8 \times 30 = 240$$

$$10(15) + 6(15) = 150 + 90 = 240$$

$$4(4 \times 15) = 4(60) = 240$$

$$1 \times 15 = 15$$

$$2 \times 15 = 30$$

$$4 \times 15 = 60$$

$$8 \times 15 = 120$$

$$16 \times 15 = 240$$

$$20(15) - 4(15) = 300 - 60 = 240$$



“Only do it this way”

$$\begin{array}{r} 3 \\ 16 \\ \times 15 \\ \hline 80 \\ +160 \\ \hline 240 \end{array}$$



Algoritma Penjumlahan

Contoh :

Anda memiliki bilangan 1,2,3,4, ..., n. Hitunglah hasil jumlah semua bilangan itu.

Algoritma Penjumlahan

Algoritma 1 (Mengkuli)

$$1 + 2 + 3 + 4 \dots + n = \dots$$



Kelebihan : gatau hehe

Kekurangan : Capek, Ada potensi gak teliti, missal $n = 1000$
gimana masih mau?

Algoritma Penjumlahan

Algoritma 1 (Mengkuli)

$$1 + 2 + 3 + 4 \dots + n = \dots$$

```
int n;  
cin>>n;  
int sum = 0;  
for(int i = 1; i<=n ; i++) {  
    sum+=i;  
}
```

Algoritma Penjumlahan

Algoritma 2 (MengCerdas)

$$1 + 2 + 3 + 4 \dots + n = \dots$$



BEROTAK SENKU

Contoh untuk $n = 10$

$$1 + 2 + 3 + 4 + \dots + 10$$

$$= (1 + 10) + (2 + 9) + (3 + 8) + \dots + (5+6)$$

$$= 11 \times 5 = 55$$

Algoritma Penjumlahan

Algoritma 2 (MengCerdas)

$$1 + 2 + 3 + 4 \dots + n = \dots$$

$$\begin{aligned}\sum_{j=1}^n j &= 1 + 2 + 3 \dots + (n-1) + n \\ &= (1+n) + ((n-1)+2) + ((n-2)+3) \dots \\ &= \frac{n(n+1)}{2}\end{aligned}$$



Algoritma Penjumlahan

Algoritma 2 (MengCerdas)

$$1 + 2 + 3 + 4 \dots + n = \dots$$

Kita langsung substitusi aja n ke rumus

$$= \frac{n(n+1)}{2}$$



Kelebihan : Cepet lah woy pakek nanya

Kekurangan : harus hafal rumusnya dikit :'

Algoritma Penjumlahan

Algoritma 2 (Mengcerdas)

$$1 + 2 + 3 + 4 \dots + n = \dots$$

```
int n;  
cin>>n;  
int sum = (n*(n+1))/2
```

Algoritmika

26. Perhatikan program di bawah ini

```
int main() {  
    int a, b, c, d, x;  
    cin >>a>>b>>c>>d;  
    a = a + a;  
    b = a + b;  
    c = a + b + c;  
    d = a + b + c + d;  
    x = a + b + c + d;  
    cout << x;  
    return 0;  
}
```

Jika program tersebut dijalankan dengan masukan 1 2 3 5 berapakah nilai x yang akan dicetak?

Jawaban: {tuliskan jawaban dalam bentuk ANGKA saja}

$$\begin{aligned}d' &= 2 + 4 + 3 + 5 \\&= 20\end{aligned}$$

$$\begin{aligned}x &= 2 + 4 + 3 + 20 \\&= \underline{\underline{35}}\end{aligned}$$

$$\begin{array}{r} \cancel{a} = 1 \\ b = 2 \\ \hline c = 3 \\ \hline d = 5 \\ \hline \end{array}$$

$$\begin{array}{r} a' = a + a \\ \hline a' = 2 \\ b' = 2 + 2 \\ \hline \underline{\underline{c'}} = 4 \\ c' = 2 + 4 + 3 \\ \hline = 9 \end{array}$$

Algoritmika

27. Perhatikan potongan program berikut!

```
if (a > b) {
    if (a > c) {
        if (d > a){
            x = d * d;
        } else {
            x = a * a;
        }
    } else {
        if (d > c) {
            x = d * d;
        } else {
            x = c * c;
        }
    }
} else {
    if (b > c) {
        if (d > b) {
            x = d * d;
        } else {
            x = b * b;
        }
    } else {
        if (d > c){
            x = d * d;
        } else {
            x = c * c;
        }
    }
}
```

Jika nilai $a=12$, $b=23$, $c=45$, dan $d=78$, berapakah nilai dari x setelah potongan program tersebut dijalankan?

Jawaban: {tuliskan jawaban dalam bentuk ANGKA saja}

Algoritmitika

```
int res = 0;  
for(int i = 1; i<=1000 ; i++) {  
    if(i%2 == 0) { habis dibagi 2 cetak 1-1000  
        res++;  
    } else if(i%3 == 0) {  
        res++;  
    } habis dibagi 3
```

Algoritmitika

```
int res = 0;
for(int i = 1; i<=1000 ; i++) {
    if(i%2 == 0) { ✓
        res++; ✓
    }
    if(i%3 == 0) { ✓
        res++; ✓
    }
}
```

Terima Kasih ☺

GWS Buat Hari ini.