BAHASA PEMROGRAMAN

Zulfikar sembiring

Definisi

- **Bahasa Pemrograman** merupakan notasi yang dipergunakan untuk mendeskripsikan proses komputasi dalam format yang dapat dibaca oleh komputer dan manusia

- **Bahasa Natural** dirancang untuk memfasilitasi komunikasi antar manusia

- **Bahasa Pemrograman** dirancang untuk memfasilitasi komunikasi antara manusia dengan komputer

Tingkatan Bahasa Pemrograman

- Bahasa Mesin (Machine Languages)
- Bahasa Rakitan (Assembly Languages)
- Bahasa Tingkat Tinggi (High Level Languages)

Bahasa Mesin

- Diawali dengan ENIAC dan EDVAC, komputer generasi pertama yang dikembangkan oleh John W. Mauclhy dan John von Neumann pada Agustus 1944
- **ENIAC** menggunakan *decimal arithmetic*
- **EDVAC** menggunakan *binary arithmetic*
- Mengalami perubahan sangat besar setelah 4 dekade sejak EDVAC

Bahasa Rakitan (Assembly)

- Dimulai sejak awal 1950
- Disebut juga sebagai *symbolic machine languages*
- Symbol yang dipergunakan untuk memfasilitasi aspek pemrograman :
  - *Mnemonic Opcodes*
  - *Symbolic Names*
  - *Memory Address Automatic Assignment*
  - *Addressing Modes*

High Level languages

- Dimulai sejak akhir 1950
- Memiliki kemampuan untuk merepresentasikan algoritma yang kompleks
- *Human-oriented readability*
- *Machine-independent*

Perbandingan

Penambahan dua buah integer

- Machine Language

```
10100101   00000001
11100101   00000010
10000101   00000011
```

- Assembly

```
LOAD A
ADD  B
STO  C
```

- High Level (contoh dengan Pascal)

```
C := A + B;
```

Sejarah Bahasa Pemrograman

- 1830 - 1840, Charles Babbage, Analytical Engine
- Programmer pertama: Ada Byron Countess Of Lovelace
- 1940, John von Neumann, Komputer pertama dengan stored programs

Era 1950 an

- FORTRAN (FORmula TRANslation), 1954~1957, IBM, John Backus, arrays, loops, if-statements
- COBOL (Common Business-Oriented Language), 1959~1960, US DOD, Grace Hopper, records, output formatting
- Algol60 (ALGOrithmic Language), 1958~1960,, structured statements, begin-end blocks, type declarations, recursion, pass-by-value parameters
- LISP (LISt Processor), akhir 1950s, MIT, John McCarthy, functions dan garbage collection
- APL (A Programming Language), akhir 1950s, Harvard University dan IBM, K. Iverson, arrays and matrices

Era 1960 an

- PL/I, 1963-1964, IBM, concurrency, exception handling
- Algol68, 1963-1968, general type system, orthogonal language
- SNOBOL (StriNg Oriented symBOlic Language), awal 1960s, Bell Labs, R. Griswold, string processing, pattern matching
- Simula67, 1965~1967, Norwegian Computing Center, Kristen Nygaard dan Ie-Johan Dahl, simulations, classes
- ISWIM, Peter Landin, functional language murni
- BASIC, 1964, Dartmouth College, John Kemeny dan Thomas Kurtz, bahasa instructional yang sederhana dan interaktif

Era 1970 an

- Pascal, 1971, Niklaus Wirth, bahasa instruksional sederhana dengan pernyataan terstruktur
- C, 1972, Bell Labs, Dennis Ritchie, type system sederhana dan runtime environment
- CLU, 1974~1977, MIT, Barbara Liskov, pendekatan konsisten untuk mekanisme abstraksi, data abstraction, control abstraction, and exception handling
- Euclid, 1976~1977, University of Toronto, abstract data types, program verification
- Mesa, 1976~1979, Xerox, module facility, exception handling, concurrency

Era 1980 an

- Ada, 1983, DOD, abstract data types, concurrency
- Modula-2, 1982, Niklaus Wirth, modules, coroutine
- Smalltak, 1980, Xerox, Alan Kay and Dan Ingalls, a complete object-oriented programming system
- C++, 1980, Bell Labs, Bjarne Stroustrup, classes, library
- Scheme, 1975~1978, MIT, Gerald Sussman dan Guy Steele, versi baru dari LISP
- ML, 1978, Edinburgh University, Robin Milner, a syntax and type checking similar to Pascal
- Prolog, 1972~1980, A. Colmerauer, mathematical logic

Era 1990 an

- Java, 1995, Gosling, bahasa pertama yang dikeluarkan dengan API yang sudah dikembangkan
- Bahasa-bahasa Scripting seperti : Perl, Tcl, Javascript, VBScript, Python, dan PHP

Era 2000 an

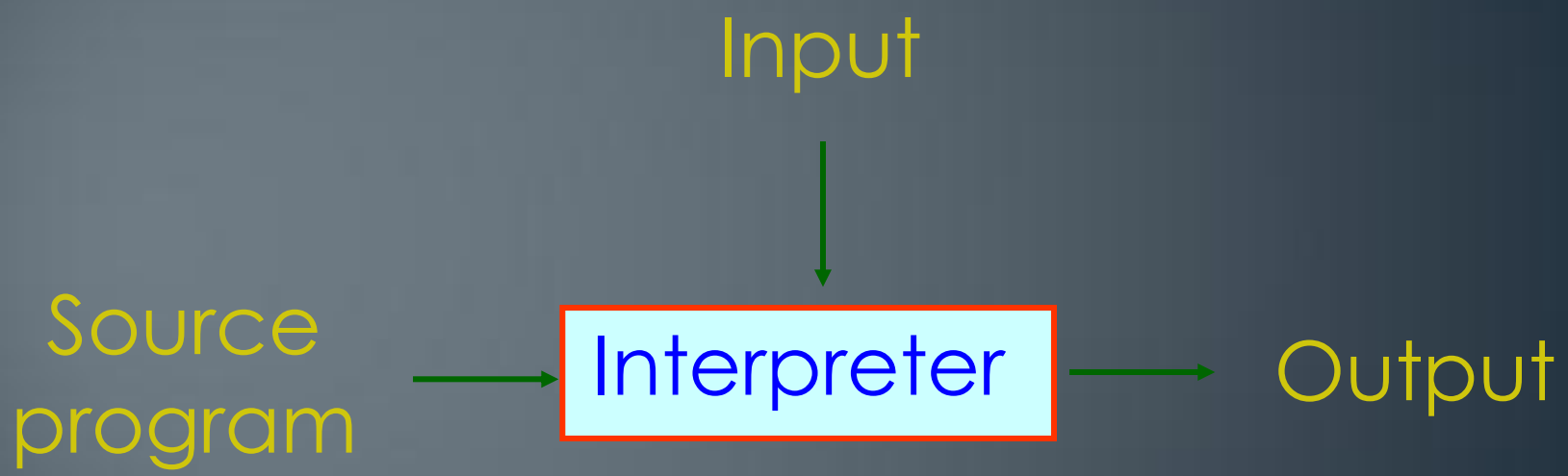
- .NET framework yang dikembangkan oleh Microsoft
- Mono yang mengadopsi .NET framework. Awalnya dikembangkan pada platform Linux

Interpreter & Kompiler

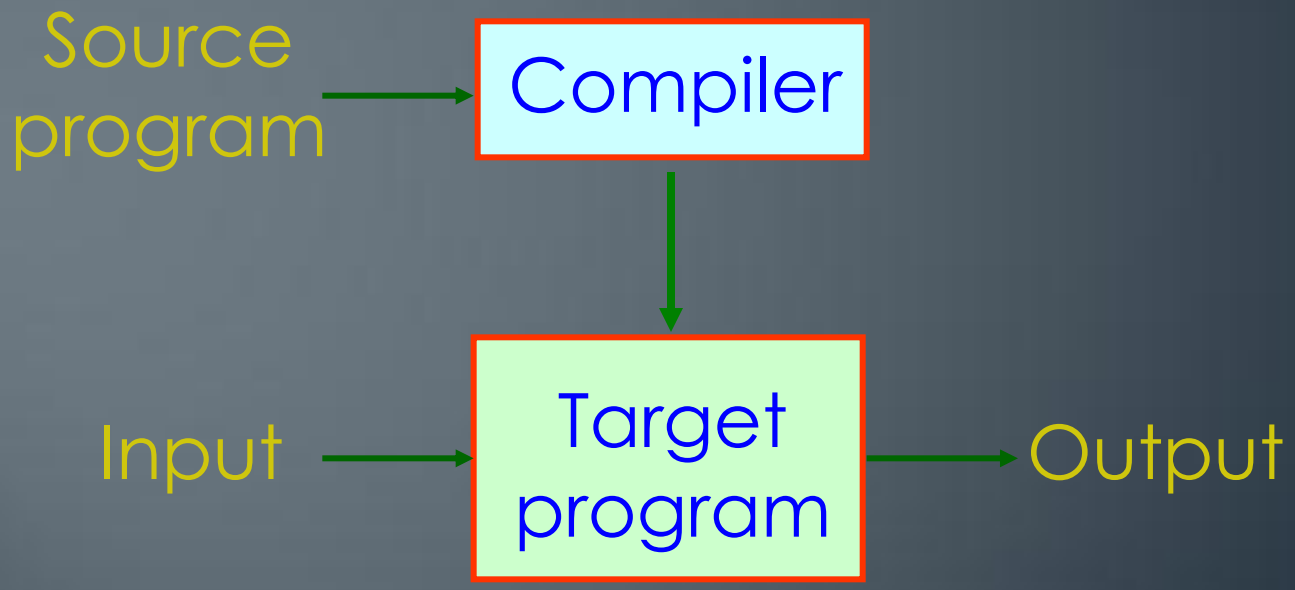
- **Interpreter** merupakan sebuah program yang *dapat mengerti* sebuah bahasa dan *mengeksekusi program* yang ditulis dengan bahasa tersebut

- **Compiler** merupakan program yang *menterjemahkan* program yang ditulis dengan sebuah bahasa *menjadi program* yang ditulis oleh bahasa lain

Interpreter



Compiler



Semantics & Syntax

- *Semantics* dari bahasa pemrograman menspesifikasikan arti dari program

- *Syntax* dari bahasa pemrograman menspesifikasikan struktur dari program

Semantics

```
If x > 2 Then
  zz:=xx**55
Else
  zz:=xx;
```

Expression Evaluating

Executing statements in TRUE condition block

Optionaly, execute statements in FALSE condition block

Mendeskripsikan bagaimana program berjalan

Syntax

```
If x > 2 Then
  zz:=xx**55
Else
  zz:=xx;
```

KEYWORD

EXPRESSION

STATEMENTS

Mendeskripsikan bagaimana program ditulis

Semantics

- Dasar mekanisme abstraksi pada bahasa pemrograman adalah penggunaan nama atau *identifiers*

- Pada kebanyakan bahasa pemrograman, variabel, konstanta dan prosedur dapat diberikan nama yang didefinisikan oleh programmer

Atribut

- Arti dari nama ditegaskan oleh atribut yang diasosiasikan oleh nama tersebut

```
const phi = 3.14;    {phi merupakan sebuah konstanta}

var x: integer;      {x merupakan sebuah variabel}

procedure Cetak;     {Cetak merupakan sebuah prosedur}
```

Binding

- Proses mengasosiasikan atribut ke nama disebut dengan *Binding*

```
const phi = 3.14;   {static binding}

var x: integer;     {static binding}

x:=2;               {dynamic binding}
```

Binding Time

- Language definition time: pada saat pendeklarasian
- Language implementation time: pada saat penggunaan
- Translation time: tipe-tipe dari variabel
- Link time: pada saat pemanggilan fungsi external
- Load time: lokasi global variabel
- Execution time: nilai dari variabel, lokasi local variabel

Deklarasi

- Deklarasi merupakan prinsip dalam menyediakan *binding*
- Umumnya diasosiasikan dengan sebuah *block*
- **Lokal**, deklarasi yang ditempatkan didalam block
- **Global**, deklarasi yang ditempatkan diluar block

Deklarasi

```
program Test;

Uses crt;

var x: integer;

procedure Cetak(y: integer);
var z: integer;
begin
   z := 2 * x;
   writeln (z + y);
end;

begin
   x := 5;
   Cetak(x);
   Readln;
end.
```

Ruang Lingkup (Scope)

```
program Test;

Uses crt;

var x: integer;

procedure Cetak(y: integer);
var z: integer;
begin
   z := 2 * x;
   writeln (z + y);
end;

begin
   x := 5;
   Cetak(x);
   Readln;
end.
```

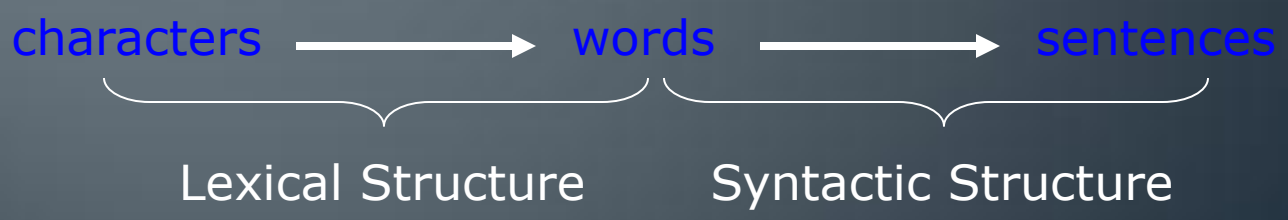
Ruang Lingkup (Scope)

```
program Test;

Uses crt;

var x: integer;

procedure Cetak(y: integer);
var z: integer;
begin
   z := 2 * x;
   writeln (z + y);
end;

begin
   x := 5;
   z := 3;
   Cetak(x);
   Readln;
end.
```

ERROR !!!

Syntax

- *Lexical Structure* menspesifikasikan bagaimana kata dibentuk dari karakter
- *Syntactic Structure* menspesifikasikan bagaimana kalimat dibentuk dari kata



Lexical Structure

- Pada lexical structure, bahasa pemrograman menggunakan *tokens* untuk membentuk *grammatical categories* yang akan membentuk blok-blok syntax
- Standar tokens:
  - *Keywords*, seperti IF, WHILE, REPEAT, dll
  - *Literals*, seperti 10 (numeric literal) atau 'A' (string literal)
  - *Special Symbols*, umumnya dipergunakan untuk membentuk operator
  - *Identifiers* , umumnya dipergunakan untuk menamai routine (prosedur & fungsi)
  - *Comments*, baris program yang tidak akan dieksekusi

Syntactic Structure

- Menggunakan notasi *Backus-Naur Form (BNF)* untuk definisi formal
- Contoh :

Binary :

```
binaryDigits → 0
binaryDigits → 1
binaryDigits → 0|1
```

Integer :

```
integer → digit|integer digit
digit → 0|1|2|3|4|5|6|7|8|9
```

Parse Tree

integer
→**integer** digit
→**integer digit** digit
→**digit** digit digit
→ 1 digit digit
→ 1 2 digit
→ 1 2 3

Sentential Form

Sentence