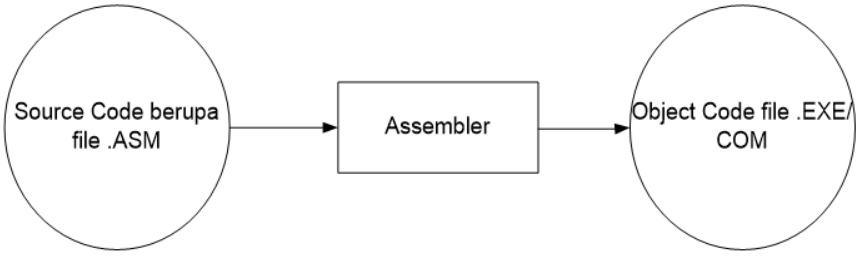
BAB 2
LANDASA TEORI

2.1 Translator

Sebuah translator melakukan pengubahan *source code/source program* (program sumber) ke dalam *target code/object code/object program* (program objek) [8]. *Source code* ditulis dalam *source language* seperti cobol, pascal, fortran sedang *object code* biasanya berupa suatu bahasa pemrograman lain atau bahasa mesin pada suatu komputer. Ada beberapa macam translator:

1. Assembler

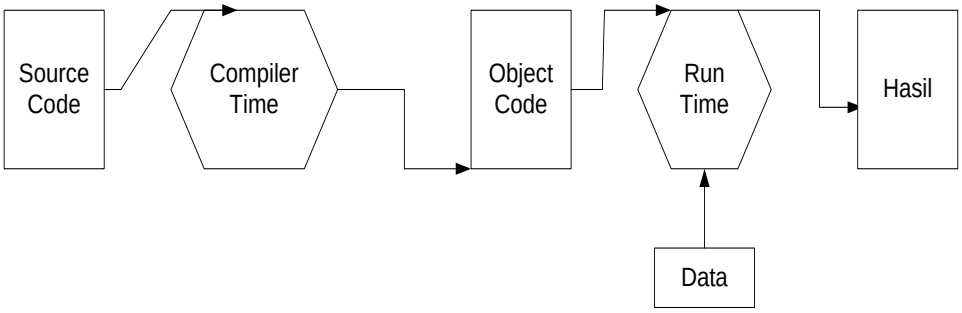
   Source code adalah bahasa assembly, object code adalah bahasa mesin. Contohnya: Turbo Assembler dan Macro Assembler [8].



Gambar 0.1 Proses Translator Assembler

2. Kompilator (*Compiler*)

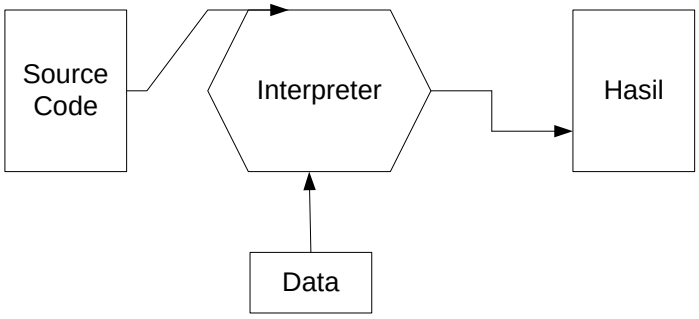
   source code dan data diproses pada saat yang berbeda. contohnya: Turbo Pascal [8], proses kompilasi dapat dilihat pada Gambar 2.2.



Gambar 0.2 Proses Kompilasi

3. Interpreter

   Interpreter tidak membangkitkan object code, hasil translasi hanya dalam bentuk internal. Contoh interpreter: BASICA/GW-BASIC, LISP, SMALLTALK. Source code dan data diproses pada saat yang sama [8]. Proses interpretasi dapat dilihat pada Gambar 2.3.



Gambar 0.3 Proses Interpreter

2.2 Algoritma

Algoritma adalah runtunan langkah-langkah untuk menyelesaikan suatu masalah [9]. Untuk masalah dengan instansiasi yang kecil, kita dapat menyelesaikan masalah itu dengan mudah dan cepat. Akan tetapi bagaimana jika instansiasi masalah berukuran besar, misalnya pada masalah pengurutan, jika n = 10000 tentu saja tidak akan mudah untuk mengurutkan data sebanyak itu. Oleh karena itu kita perlu menuliskan langkah-langkah pengurutann sehingga dapat dijalankan oleh sebuah

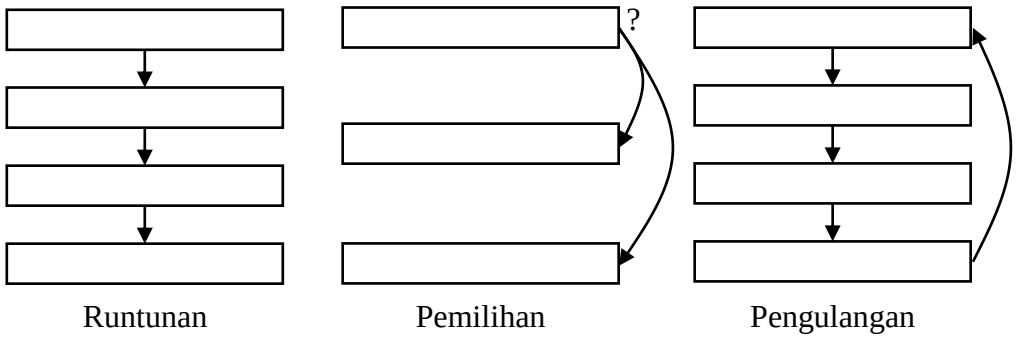
pemproses (komputer, robot dan sebagainya) untuk menghasilkan solusi instansiasi masalah pengurutan.

2.2.1 Notasi Algoritmik

1. Menyatakan langkah-langkah algoritma dengan untaian kalimat deskriptif.
   Dengan menggunakan untaian kalimat ini, dapat dijabarkan pada setiap proses langkah-langkahnya secara gamblang [9].
2. Bagan Alur
   Bagan alur (*flowchart*) adalah notasi algoritmik yang menggunakan sekumpulan bentuk-bentuk geometri untuk menggambarkan proses, misalnya bentuk persegi panjang menyatakan proses, bentuk intan menggambarkan pernyataan kondisi [9].
3. *Pseudocode*
   Pseudocode merupakan format penulisan algoritma yang menggabungkan bahasa alami dan bahasa pemrograman [9].

2.2.2 Struktur Dasar Algoritma

Algoritma merupakan deskripsi langkah-langkah pelaksanaan suatu proses. Setiap langkahnya dinyatakan sebagai *statement* atau istilah lainnya instruksi [9]. Algoritma dapat dibangun dari tiga buah konstruksi atau struktur dasar, yaitu runtunan, pemilihan atau percabangan, dan pengulangan.



Gambar 0.4 Struktur Dasar Algoritma [9]

1. Runtunan

   Runtunan merupakan struktur paling dasar yang berisi rangkaian instruksi yang diproses secara berurut (sekuensial), satu per satu, mulai dari instruksi pertama sampai instruksi terakhir [9].

2. Pemilihan

   Pernyataan pemilihan adalah cara yang digunakan untuk pengambilan keputusan dari kondisi yang ada. Contohnya, jika sepeda motor mogok, maka gunakan angkot untuk ke kampus. Pernyataan tersebut dapat ditulis dalam pernyataan pemilihan (*selection-statement*), atau disebut juga pernyataan-kondisional [9].

3. Pengulangan

   Pengulangan adalah proses untuk melaksanakan perintah sebanyak n kali atau sampai kondisi terpenuhi [9].

2.3 Source Code

Source code pada ilmu komputer merupakan kumpulan-kumpulan perintah yang digunakan untuk menyelesaikan masalah yang ditulis dalam bahasa yang dapat dipahami oleh komputer [9].

2.4 Bahasa Pemrograman *Pascal*

Pascal adalah bahasa pemograman komputer tingkat tinggi (high level language) dan orientasinya pada segala tujuan [10]. Bahasa pemrograman pascal banyak digunakan dalam dunia pendidikan. Karena pascal relatif mudah dibaca dan notasinya mirip bahasa inggris standar [9] seperti *begin, end, write,* dan *read*. Hal ini berbeda dengan bahasa turunan C seperti C++, C#, PHP atau JavaScript yang lebih banyak menggunakan simbol-simbol.

Pada Pascal, pengenal (*identifier*) adalah nama yang dapat diberikan pada suatu elemen program, dapat berupa konstanta, variabel, fungsi, suatu prosedur, maupun suatu program [11]. Beberapa aturan pada *identifier* yaitu [11]:

1. Nama pengenal harus diawali dengan karakter huruf.
2. Karakter kedua dan selanjutnya dapat berupa kombinasi angka dan huruf, tetapi tidak boleh menggunakan karakter khusus seperti ?, #, dan sebagainya. Namun ada beberapa versi Pascal yang menerima garis bawah '_' sebagai karakter penyusun nama pengenal.
3. Panjang karakter yang digunakan sebagai pengenal bisa sembarang, tapi dalam beberapa versi Pascal hanya mengenal delapan karakter awal, karakter sembilan dan seterusnya diabaikan.
4. Beberapa karakter sudah digunakan oleh Pascal untuk tujuan tertentu, sehingga tidak dapat dipakai sebagai nama pengenal. Nama-nama ini disebut kata khusus.
5. Beberapa nama yang disebut pengenal standar juga telah mempunyai arti khusus, tetapi jika didefinisikan ulang maka dapat menjadi nama pengenal. Jika pengenal standar digunakan sebagai pengenal biasa maka arti khususnya tidak akan dipergunakan.

2.5 *Natural Language Processing*

Natural Language Processing (NLP) adalah bagian dari Artificial Intelligence yang mengupayakan agar komputer dapat memahami dan memberikan output dalam bentuk bahasa manusia[11]. *Natural Language Processing* terdiri dari 3 komponen utama, yaitu *knowledge base*, *inference engine*, dan *user interface.* Salah satu tantangan pada pemrosesan bahasa alami yaitu pemilihan arti dari kata yang memiliki makna lebih dari satu, seperti kata 'genting', kata 'genting' bisa bermakna 'atap' dan dapat juga bermakna 'gawat' sesuai dengan bentuk kalimatnya [12]. Penelitian yang dapat menangani kasus kata yang mempunyai lebih dari satu makna adalah dengan cara Part of Speech Tagger, seperti yang dilakukan oleh Purnamasari dan Suwardi [13].

2.6 *Grammer*

Grammer adalah sekumpulan aturan yang mengeksplorasi bentuk dan struktur kalimat yang bisa digunakan dalam suatu bahasa [14]. Ada dua konsep penting yang berkaitan dengan *grammer* :

1. Morphology yang mempelajari pembentukan kata-kata, struktur dan hubungan antar kata [15].
2. *Syntax* yang mempelajari struktur kalimat, hubungan antara unit kalimat, struktur internal frasa dan hubungan antar kata yang memberi makna untuk kalimat [15].

2.7 *Case Folding*

Case folding merupakan proses menyeragamkan huruf pada teks masukan menjadi huruf kecil (*lowercase*) atau huruf kapital (*uppercase*). Dalam penelitian ini penyeragaman huruf digunakan untuk mengubah semua huruf pada teks masukan menjadi huruf kecil.

2.8 *Filtering*

Filtering biasanya dilakukan pada dokumen untuk menghapus beberapa *stop word* [16]. Pada penelitian ini karakter yang diperbolehkan dalam penelitian ini yaitu ‘a’ sampai ‘z’, ‘0’ sampai ‘9’, koma (‘,’), titik (‘.’), garis bawah(‘_’), dan spasi.

2.9 *Scanning*

Scanning merupakan tahap memilah teks masukan menjadi token-token berdasarkan kelasnya. Pada penelitian ini, tahap *scanning* menerima masukan *stream* kata yang kemudian memilah teks masukan menjadi satuan leksik atau token [8]. *Scanner* juga membaca karakter input satu per satu untuk menentukan unit kata primitif, atau "token", dari setiap perintah. Besaran pembangun token atau leksik meliputi hal-hal sebagai berikut.

1. *Identifier*

Identifier dapat berupa *keyword*. *Keyword* merupakan kata kunci yang sudah didefinisikan oleh suatu bahasa [8].

2. Nilai Konstanta

   Nilai konstanta merupakan konstanta yang ada pada suatu teks. Dapat berupa nomor, string, dan sebagainya [8].

3. Operator dan *Delimiter*

   Operator dapat berupa operator aritmetika atau operator logika. Delimiter adalah pemisah atau pembatas, misalnya titik, koma, spasi, dan sebagainya [8].

2.10 *Parsing*

Parsing adalah proses menentukan bagaimana string terminal dapat dihasilkan oleh *grammer* [17]. Metode parsing digolongkan menjadi 2 yaitu :

1. *Top Down*

   Metode ini melakukan penurunan dari *root*/puncak menuju *leaf*/daun. Metode *top down* meliputi *Backtack/backup* dan *No backtrack.*

2. *Bottom Up*

   Metode ini melakukan penurunan dari *leaf*/daun menuju root.

2.11 Pohon Sintaks

Pohon sintaks berguna untuk menggambarkan bagaimana memperoleh suatu *string* dengan cara menurunkan simbol-simbol variabel menjadi simbol-simbol terminal [8]. Setiap simbol variabel akan diturunkan sampai tidak ada lagi simbol variabel atau hanya menyisakan simbol terminal. Proses penurunan atau *parsing* bisa dilakukan dengan cara sebagai berikut [8].

1. Penurunan paling kiri (*leftmost derivation*): simbol variabel paling kiri yang akan diperluas terlebih dahulu.

   Misal, terdapat tata bahasa bebas konteks:

   $$S \rightarrow aAS \mid a$$

A → SbA | iba

Untuk memperoleh untaian ‘aabbaa’ dari tata bahasa bebas konteks diatas maka penurunannya sebagai berikut.

S => aAS => aSbAS => aabAS => aabbaS => aabbaa.

2. Penurunan paling kanan (*rightmost derivation*): simbol variabel atau non terminal paling kanan yang akan diperluas terlebih dahulu.

   Misal, terdapat tata bahasa bebas konteks:

   S → aAS | a

   A → SbA | ba

   Untuk memperoleh untaian ‘aabbaa’ dari tata bahasa bebas konteks diatas maka penurunannya sebagai berikut.

   S=> aAS => aAa => aSbAa => aSbbaa => aabbaa.

2.12 Pengujian Akurasi

Pengujian yang dilakukan dalam penelitian ini menggunakan pengujian akurasi. pengujian dilakukan untuk mengetahui nilai akurasi dari sistem yang dibangun. Pengujian dilakukan dengan cara membandingkan hasil translasi *source code* dari sistem dengan *source code* yang diharapkan. Pada penelitian ini rumus yang digunakan untuk pengujian akurasi sebagai berikut.

$$Akurasi = \frac{Jumlah\ hasil\ source\ code\ benar}{Jumlah\ keseluruhan\ data\ uji} \times 100\%$$

Tabel 0.1 Contoh Pengujian

No.	Kode Teks	*Source Code* yang Diharapkan	*Hasil Source* Code Dari Sistem	Hasil
1	Buat program uji1. buat variabel i dengan tipe data integer. buat	program uji1; var	program uji1; var	Benar

	perulangan untuk iterasi i bernilai 1 sampai 10 lakukan Tampilkan “benar”.	i : integer;
		begin
		for i := 1 to 10 do
		begin
		writeln (‘ benar ‘) ;
		end ;
		end.
			i : integer;
			begin
			for i := 1 to 10 do
			begin
			writeln (‘ benar ‘) ;
			end ;
			end.	

2	Buat program uji2. buat variabel x dengan tipe data integer. Buat perulangan ulangi tampilkan hello world x bernilai x ditambah 1 sehingga x lebih besar dari 10.	program uji2 ;
		var
		x : integer;
		begin
		repeat
		begin
		writeln (‘ hello world ‘);
		x := x + 1 ;
		end ;
		until x > 10 ;
		end.
			program uji2 ;
			var
			x : integer;
			begin
			repeat
			begin
			writeln (‘ hello ‘, ‘ world‘);
			x := x + 1 ;
			end ;
			until x > 10 ;
			end.
				Salah

2.13 Model Perangkat Lunak

2.13.1 Flowchart

flowchart adalah notasi algoritmik yang menggunakan sekumpulan bentuk-bentuk geometri untuk menggambarkan proses, misalnya bentuk persegi panjang menyatakan proses, bentuk intan menggambarkan pernyataan kondisi [9]. Tujuan utama dari penggunaan *flowchart* adalah untuk menggambarkan suatu tahapan penyelesaian masalah secara sederhana, terurai, rapi, dan jelas dengan menggunakan simbol-simbol yang standar. Tahap penyelesaian masalah yang disajikan harus jelas,

sederhana, efektif, dan tepat. Dalam pembuatan *flowchart* dikenal dua model, yaitu sistem *flowchart* dan program *flowchart* [18].

2.13.2 DFD (*Data Flow Diagram*)

Diagram aliran data merupakan model dari sistem untuk menggambarkan pembagian sistem ke modul yang lebih kecil [19]. DFD sering digunakan untuk penggambaran pada suatu sistem yang telah ada atau sistem yang baru yang akan dikembangkan secara logika dan menjelaskan aliran data dari mulai pemasukan data sampai dengan keluaran data tingkatan diagram arus data mulai dari diagram konteks yang menjelaskan secara umum suatu sistem atau batasan sistem dari level 0 diturunkan menjadi level 1 sampai sistem dapat tergambarkan secara terperinci.

Salah satu dari keuntungan menggunakan diagram aliran data adalah memudahkan pemakai (*user*) yang kurang menguasai bidang komputer untuk mengerti sistem yang akan dikerjakan [19]. Beberapa komponen yang digunakan pada DFD adalah:

1. Entitas Luar (*Boundary*)

   Entitas luar adalah kesatuan di lingkungan luar sistem yang dapat berupa orang, organisasi atau sistem lainnya yang berada di lingkungan luarnya yang akan memberikan masukan (*input*) atau menerima keluaran (*output)* dari sistem [20].
2. Arus Data (*Data Flow*)

   Arus data diberi simbol suatu panah. Arus data mengalir diantara proses (*process*), simpanan data (*data store*) dan kesatuan luar (*external entity*) [20].
3. Proses (*Process*)

   Proses adalah kegiatan atau kerja yang dilakukan oleh orang, mesin atau komputer dari hasil suatu arus data yang masuk ke dalam proses untuk menghasilkan arus data yang akan keluar dari proses [20].
4. Penyimpanan Data (*Data Store*)

Penyimpanan data adalah simpanan dari data yang dapat berupa *file* atau *database* di sistem komputer, arsip atau catatan manual, agenda atau buku. Simpanan data DFD dapat di simbolkan dengan sepasang garis horizontal [20].

2.14 PHP (*Personal Home Page*)

PHP (*Hypertext Preprocessor*) yaitu bahasa pemrograman *web server-side* yang bersifat *opensource*. *PHP* merupakan *script* yang terintegrasi dengan *HTML* dan berada pada *server* (*server side HTML embedded scripting*). *PHP* adalah *script* yang digunakan untuk membuat halaman *website* yang dinamis. Dinamis berarti halaman yang akan ditampilkan dibuat saat halaman itu diminta oleh *client*. Mekanisme ini menyebabkan informasi yang diterima *client* selalu yang terbaru / *up to date*. Semua *script PHP* dieksekusi pada *server* di mana *script* tersebut dijalankan [21]. Pada penelitian ini bahasa pemrograman yang dipakai yaitu *PHP*, karena bahasa pemograman *PHP* itu *open source, dynamic*, dan sederhana [21].

2.15 *Software* Pendukung

2.15.1 XAMPP

XAMPP merupakan sebuah aplikasi *web server*. *Web Server* sendiri adalah sebuah aplikasi tempat Anda menyimpan file – file maupun data – data untuk membuat *website*. Juga sering diartikan sebagai penerima permintaan berupa halaman *client* dan mengirimkan kembali hasil yang diminta dalam bentuk halaman *web* [21]. XAMPP sangat mudah digunakan bagi pemula, karena pada proses instalasinya tidak memerlukan konfigurasi Apache, PHP dan MySQL secara manual, XAMPP melakukan instalasi dan konfigurasi secara otomatis.