BAIK–PROGRAMMING LANGUAGE BASED ON INDONESIAN LEXICAL PARSING FOR MULTITIER WEB DEVELOPMENT

Haris Hasanudin

Computer Science, Gunma University, Gunma Daigaku, 4-2 Aramaki-Machi, Maebashi-shi, Gunma 371-8510, Japan

Email: hariscom@ieee.org

Abstract

Business software development with global team is increasing rapidly and the programming language as development tool takes the important role in the global web development. The real user friendly programming language should be written in local language for programmer who has native language is not in English. This paper presents our design of BAIK (Bahasa Anak Indonesia untuk Komputer) scripting language which syntax is modeled with Bahasa Indonesian for multitier web development. Researcher propose the implementation of Indonesian Parsing Engine and Binary Search Tree structure for memory allocation of variable and compose the language features that support basic Object Oriented Programming, Common Gateway Interface, HTML style manipulation and database connection. Our goal is to build real programming language from simple structure design for web development using Indonesian lexical words.

Keywords: database access, indonesian lexical parser, multitierweb development, scripting language

1. Introduction

The need of programmer job for building web based business software with global team in the cloud computing era is increasing rapidly. However, the number of global programmer is not growing as fast as market demand. The web global software development with involving developers from many countries faces the language barrier not only in spoken communication between members, but also in programming code level.

The programmers should focus not only on program algorithm, but also should write and understand the English based programming languages. In this paper, researcher explain the scripting language which is suitable for building multitier web application using lexical sentence in Indonesian Language. The main concept of this local scripting language is easy to use for everyone and more easily to be introduced to the programmer since junior or senior high school level.

We call our scripting language as BAIK (Bahasa Anak Indonesia untuk Komputer), which means language for computer by Indonesian kid with lexical syntax is written in Indonesian. This scripting language supports basic components of programming language such as basic arithmetic operation, logical condition, array, text file handling, and simple object oriented programming. For multitier web programming,
this language is able to be used in server side programming and has not only web interaction via CGI (Common Gateway Interface) [1], but also database connection. This scripting language was developed using GNU C language [2][3] and CGI programming was tested using Apache [4] web server in Windows and Unix/Linux environment.

The design of BAIK language can be divided into three parts: The first is Indonesian Parsing Engine (IPE), which read the source code from text file and determine the language word in Indonesian. Since Indonesian syntax uses alphabetic character, the standard C language functions are used to recognize Indonesian words. The second is Memory Allocation of Variables (MAV). To put the value of variable, memory allocation using Binary Search Tree model [5][6] is used as data structure. List structure of memory allocation is used to store the array of string and an object. The last is Language Function (LF), which provides the capability to process specific functions such as mathematical Sin and Cos functions. LF depends on the standard C library or other extern C library that can be adopted into BAIK language.

2. Methodology

The first target in design of BAIK language is to provide console based real scripting language for multtier web development in multi platforms. The implementation emphasizes on using Indonesian word as language word in processing the data of web application. The Graphic User Interface (GUI) for desktop application development is out of our implementation at this time. In order to help web development, BAIK language there are some functions to simplify writing HTML tags.

The flow of code reading. To read each character inside source code in BAIK and translate the group of characters as a variable or a value, researcher keep the current position pt and the back position back_pt of code reading as shown in figure 1.

BAIK interpreter starts by reading source code from text file. BAIK reads the entire source code into a variable *source with calloc memory allocation function. This memory allocation is assumed to be enough to store as big as file size of source code. In parsing [7-9] lexical word inside source code by IPE, the current parsing position and the last position are memorized in data type called S_ADDR and noticed by struct in standard C language as follow in figure 2.

```
typedef struct
{ int pt;               /* current position of parsing*/
  int back_pt;      /* last position of parsing */
  char *source;    /*source code */
} S_ADDR;
```

Figure 2. BAIK interpreter starts by reading source code from text file.

The data type for BAIK variable recognized while parsing is stored into data type called LEX_T as follow in figure 3:

```
typedef struct
{ /* type of lexical parse */
  int type;
  union{
    int num;
    double dblnum;
    char ident[MAX_IDENT_LEN];
    char string[MAX_STRING_LEN];
    char symbol;
    char array_str[MAX_STRING_LEN];
    char array_name[MAX_STRING_LEN];
    char array_idx[MAX_STRING_LEN];
    char object_str[MAX_STRING_LEN];
    char object_name[MAX_STRING_LEN];
  } detail;
} LEX_T;
```

Figure 3. Data type for BAIK variable.

If the parsing result is substitution, the value and additional information of BAIK variable is stored into data type called VAL_LABEL by MAV. If the parsing result is the name of variable itself (ident), MAV will read the value from memory.

In case the parsing result is a function, LE will call procedure that contains specific functions. BAIK language will release all memory allocation when program ended. BAIK language also has HAPUS keyword to remove the value of...
variable by setting NULL and releasing memory allocation.

Finite state machine of BAIK parsing. We can derive the finite state machine [9][10] of parsing process of IPE into 12 main states as shown in figure 4. We define the state 0 in order to indicate the end of parsing process of a word or a symbol, and the state 1 for indicating the beginning. Parsing a digit at the first character, will create two states: the first is an integer state and the second is decimal state which has dot symbol inside digit parsing. Parsing a string will produce also two states for reading the first and the last of double quotes (0x22) symbols between alphabetic characters. The line with first character is the sharp (#) will be parsed into comment state.

If the first character is an alphabetic one, there are four states can be achieved: a group of character which has brackets [ ] symbol at the tail, will be read as an array state. Character group with hyphen and more than symbol (<->) will be interpreted as an object state. If the object state has bracket () symbol, it will be recognized as an object function state. Alphabetic character group without any symbol will be accepted as the state of variable name. The other parsing result will create a state for mathematical expression symbol or the end of source code EOF symbol.

Design of variable handling. A variable in BAIK is a collection of different type of data type, called VAL_LABEL and noticed by struct in standard C language. When parsing a word of a variable, the real type is determined based on the parsing result of a word, which can be integer numeric, floating numeric, string, array or object. The real type is also memorized into VAL_LABEL. By this approach, the users do not need to define the type of variable. In other word, variable in BAIK language has automatic data type in design. BAIK design uses only one Binary Search Tree to allocate all variables except an object. Hence, all variables have global scope that can be referred from entire execution flow.

```c
typedef struct _val_label
{
    struct _val_label *left; /* left pointer of tree struct */
    struct _val_label *right; /* right pointer of tree struct */
    char ident[MAX_IDENT_LEN]; /* var or lbl name */
    int datatype;           /* type of variable*/
} VAL_LABEL;
```

Figure 6. Implementation of VAL_LABEL in C language using pointer *left and *right.
The binary tree structure of variable allocation VAL_LABEL is shown in figure 5. Implementation of VAL_LABEL in C language using pointer *left and *right is shown as follow in figure 6. The type of variable is saved in datatype parameter. The main types of parameter are listed as follow in table I.

![Binary search tree structure of variable.](image)

Figure 5. Binary search tree structure of variable.

Tree structure of MAV. The main design of variable handling in BAIK language is Binary Search Tree structure of MAV. This tree structure uses static variable *root to build binary tree which point to node that produced by operation of ValLabel function.

If value of flag is VAL_FLAG_START, ValLabel creates new binary searching tree *node. All components of VAL_LABEL are initialized and sufficient memory is allocated. When flag is VAL_FLAG_SEARCH_W, a data as a new node is added into binary searching tree *node. When VAL_FLAG_SEARCH_R is given into flag, ValLabel searches a data from binary searching tree *node. By VAL_FLAG_END, all data in *node will be erased. ValLabel function is defined as follow in figure 7.

![Tree structure of MAV.](image)

Figure 7. Tree structure of MAV.

### Table 1

<table>
<thead>
<tr>
<th>Type</th>
<th>Kind of variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Integer</td>
</tr>
<tr>
<td>1</td>
<td>Double/decimal</td>
</tr>
<tr>
<td>2</td>
<td>File</td>
</tr>
<tr>
<td>3</td>
<td>String</td>
</tr>
<tr>
<td>4</td>
<td>Socket of TCP connection</td>
</tr>
<tr>
<td>5</td>
<td>Socket of UDP connection</td>
</tr>
<tr>
<td>6</td>
<td>Array of integer</td>
</tr>
<tr>
<td>7</td>
<td>Array of double/decimal</td>
</tr>
<tr>
<td>8</td>
<td>Array of string</td>
</tr>
<tr>
<td>9</td>
<td>Function</td>
</tr>
<tr>
<td>30</td>
<td>Object definition</td>
</tr>
<tr>
<td>31</td>
<td>Object instance</td>
</tr>
<tr>
<td>32</td>
<td>Function of object</td>
</tr>
<tr>
<td>33</td>
<td>Array of object</td>
</tr>
</tbody>
</table>

3. **Results and Analysis**

**BAIK** language is in single thread execution. To execute **BAIK** language as a console program, researcher simply to write baik with text file source code file as a parameter. The illustration to execute test01.ina source code in Unix/Linux environment is as follow in figure 8.

![Illustration to execute test01.ina source code in Unix/Linux environment.](image)

Figure 8. Illustration to execute test01.ina source code in Unix/Linux environment.

Storing a value into a variable is gained by using equal (=) symbol for any data type as follow in figure 9:
Figure 9. Storing a value into a variable.

<table>
<thead>
<tr>
<th>TABLE II</th>
<th>ARITHMETIC OPERATION OF BAIK LANGUAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operator</td>
<td>Operation</td>
</tr>
<tr>
<td>+</td>
<td>Addition</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
</tr>
<tr>
<td>*</td>
<td>Multiply</td>
</tr>
<tr>
<td>/</td>
<td>Division</td>
</tr>
<tr>
<td>%</td>
<td>Modulus</td>
</tr>
<tr>
<td>^</td>
<td>BUKAN</td>
</tr>
<tr>
<td>DAN</td>
<td>AND</td>
</tr>
<tr>
<td>ATAU</td>
<td>OR</td>
</tr>
<tr>
<td>==</td>
<td>Equal</td>
</tr>
<tr>
<td>&lt;&gt;</td>
<td>Not Equal</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;</td>
<td>More than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or Equal</td>
</tr>
<tr>
<td>=&gt;</td>
<td>More than or Equal</td>
</tr>
</tbody>
</table>

To print the value of a variable to standard output, researcher use keyword TULIS as follow in figure 10:

TULIS variable1, variable2, variable3, ...

Figure 10. Keyword TULIS.

To delete a value of a variable, researcher use keyword HAPUS as follow in figure 11:

HAPUS variable

Figure 11. Keyword HAPUS.

In BAIK language, program code is written per line without line number. Writing among keywords or between keyword and variable is separated by space character. A line with first character is sharp ‘#’ will be interpreted as comment line. BAIK language supports basic arithmetic operation as follow in table II. BAIK language supports basic logical operations such as IF condition, WHILE condition, FOR condition, and SWITCH condition. First, IF condition. To make logical choosing if condition is valid, researcher use keywords KALAU and MAKA. When condition is not satisfied, researcher use keyword LAINNYA to refer ELSE condition. We close this IF condition by using keyword AKHIRKALAU (figure 12). Those all will realize IF, THAN, ELSE and ENDIF statement.

To print the value of a variable to standard output, researcher use keyword TULIS as follow in figure 10:

TULIS variable1, variable2, variable3, ...

Figure 10. Keyword TULIS.

To delete a value of a variable, researcher use keyword HAPUS as follow in figure 11:

HAPUS variable

Figure 11. Keyword HAPUS.

In BAIK language, program code is written per line without line number. Writing among keywords or between keyword and variable is separated by space character. A line with first character is sharp ‘#’ will be interpreted as comment line. BAIK language supports basic arithmetic operation as follow in table II. BAIK language supports basic logical operations such as IF condition, WHILE condition, FOR condition, and SWITCH condition. First, IF condition. To make logical choosing if condition is valid, researcher use keywords KALAU and MAKA. When condition is not satisfied, researcher use keyword LAINNYA to refer ELSE condition. We close this IF condition by using keyword AKHIRKALAU (figure 12). Those all will realize IF, THAN, ELSE and ENDIF statement.

Second, WHILE condition. We use phrases SELAGI, LAKUKAN and BAIKLAGI to interpret loop operation: WHILE, DO, ENDWHILE (figure 13).

To jump outside from WHILE loop operation even the condition is still true, researcher use phrase KELUARSELAGI (figure 14).

Third, FOR condition. To make logical iteration with start and stop condition values, researcher use keywords UNTUK, ULANG and LAGI. Those will interpret loop operation: FOR, REPEAT, and ENDFOR (figure 15).

To jump outside from FOR loop operation even the condition is still true, researcher use phrase KELUARUNTUK (figure 16):
Fourth, SWITCH condition. We use phrases PILIH, SAMA and AKHIRPILIH to interpret election in choice. In this choice condition, logical flow will execute code when variable is the same with the given value as option. Those keywords will realize SWITCH, CASE, ENDCASE statement (figure 17).

Implementation of array in BAIK language is using basic one dimension array so researcher called UNTAIAN. Researcher need to determine the size of array in definition by using bracket symbol ‘[’ and ‘]’. To get the value of array size, researcher use keyword PANJANG.

Here is the illustration of using array in BAIK language:

```
Untaian[5]
p[0] = 1.5
p[1] = 3.0
L = panjang
```

Function or subprocedure is defined using keyword FUNGSI. The return value of function is given by keyword BAIK. To call a function, researcher need to put symbol ampersand ‘&’ before the function name (figure 18). The usage pattern of function Calling as follow in figure 19.

For flexible data handling by using reuse pattern of source code and perform modern programming style, BAIK language has basic capability of object oriented programming. The basic object definition, instance, cloning, and removing have been adopted.

An Object is composed by any parameters and one or many functions. An Object can be defined by keyword DEFINISIBENDA and at least has one function which name is the same with object name. To write function inside object definition, researcher use FUNGSI keyword.

```
PILIH variabel
{ 
  SAMA value-A
    Command code A
  AKHIRPILIH
  SAMA value-B
    Command code B
  AKHIRPILIH
    ...
  SAMA value-N
    Command code N
  AKHIRPILIH
}
```

Fig. 17. SWITCH condition.

In main program, researcher use keyword BENDA for defining an object parameter. To make instance of an object, keyword BENDABARU is used. To remove an object, researcher use keyword HAPUS.

To call a variable or a function of object from main program, researcher use referring format as follows:

```
mainparam->objparam
mainparam->funcname
```

where, mainparam is object variable name in mainprogram, objparam is variable name inside object definition, funcname is function name inside object definition.

```
FUNGSI functionname (param1, param2, ...)
{
  command code
  ...
  BAIK returnval
}
```

Figure 18. Function or subprocedure is defined using keyword FUNGSI

```
Param = &funcname (param1, param2, ...)
```

Figure 19. Function calling.

Object definition as follow in figure 20. Object variable and instanceas follow in figure 21. Object variable cloning as follow in figure 22. Object variable removing as follow in figure 23. There is a concept of variable scope in object definition. By default, a variable in object will have global scope. However, researcher can determine variables with local scope that means valid only inside corresponded object by using keyword LOKAL.

Figure 24 and 25 is the example of using object oriented style of BAIK language for general business application in order to record working time of employee. In this example, researcher define DataPersonel object which has nama and no_identitas variables. In main program, researcher use pegawai1 and pegawai2 for object variables and make an instance from DataPersonel object. The filename of source code is assumed as mypersonel.ina.

```
DEFINISIBENDA objectdefname[
  objparam1, objparam2,
  ...
  FUNGSI objectdefname (param1, param2, ...)
  {
    ...
  }
  FUNGSI otherfuncname (paramx, paramy, ...)
  {
    ...
  }
]
```

Figure 20. Object definition.
The result of code executing is as follow in figure 26. To support multitier web programming, BAIK has capabilities of Common Gateway Interface (CGI) for data transfer interface, default HTML layout for menu design interface and database access interface.

The interface of BAIK for passing the data through CGI are input interface and output interface. Input Interface use POST method of CGI to receive data inputted from HTML page by stating POST_CGI keyword at the first line of program. BAIK language is still able to use GET method of CGI by reading environment parameter of Operating System.

Output Interface consists of simply to write to standard output (The TULIS keyword is used to print HTML tags message directly into standard output), keywords for simplifying writing HTML tags (We create a collection keywords KERTASWEB_xxx to simplify writing HTML tags with basic menu layout), keywords for simplifying drawing directly onto HTML page (Collection keywords KANVASWEB_xxx are created to simplify using JavaScript Vector development). BAIK language uses CGI which is a standard interface for writing programs for data interacting between web server side program and a client running a web browser.

```python
# ######################################
# Program Utama - Main Program
# ######################################
Tulis "Tes Object Oriented Program dengan BAIK\n"
Benda pegawai1
Benda pegawai2
nama1 = "nama1"
nama2 = "nama2"
Tulis "Pemasukan data absenpersonel\n" pegawai1 = BendaBaruDataPersonel
pegawai1->awalan(nama1,"NIP0000001")
pegawai1->tanggal = "01/02/2009"
pegawai1->setJamMasuk("10:00")
Tulis "Hasil pemasukan data ", nama1, " telah diregistrasi\n" setelah registrasi\n" pegawai1->nama, "
Tulis "tanggal: ", pegawai1->tanggal, "\n"
Tulis "------------------------------\n"
Tulis "Pemasukan data absenselesai\n"
```

Figure 25. Main program object oriented style of BAIK language for general business application.

CGI for data transfer interface. BAIK language was designed to provide an input and output interface for Internet based software
Default HTML layout for menu design of web page interface. Serving business application via web interface requires at least one menu layout of HTML pages. BAIK language was designed to provide default HTML layout for menu design in web page. This HTML layout will be created by keyword SEDIA_KERTASWEB and has four main components: Menu area, Copyright area, Title area, and Main area as shown in figure 27.

Figure 26. Result of code executing.

Figure 28 is the example code to produce and use the standard HTML layout in a web page. The first step is to determine the real file location of HTML style will be placed and the main color of HTML style by using keyword SEDIA_KERTASWEB. The second step is to define the beginning and the end of HTML page by using KERTASWEB_AWAL and KERTASWEB_AKHIR. In third step, researcher determine the alias address of URL path for HTML style by using KERTASWEB_MODEL.

```c:aik
warna_halaman = "birumuda"
lokasi_filestyle = "C:/Program Files/Apache Group/Apache/style"
SEDIA_KERTASWEB lokasi_filestyle, warna_halaman
KERTASWEB_AWAL "KORAN WEB HARIAN UTAMA"
lokasi_style = "/style"
KERTASWEB_MODEL lokasi_style
Kertasweb_Menu_Awal
Untuk(A=1; A<9; A=A+1) ulang
kalau A==1 maka
Nama = "BeritaUtama"
Lainnya
Nama = Gabung "BeritaKe-", A
akhir
Deskripsi = gabung "Halaman ", A
Halaman = gabung "hal", A , ".html"
Kertasweb_MenuNama, Deskripsi, Halaman
Lagi
Kertasweb_Hakcipta "Copyright kertasweb - BAIK 2009"
Kertasweb_Menu_Akhir
Judul = "KORAN WEB HARIAN UTAMA"
Nama = "BERITA UTAMA"
Deskripsi = "WAKTU"
ISIWEB_AWAL
ISIWEB_TITEL Judul, Nama, Deskripsi
tulis "<h2>Berita pertama</h2>
tulis "iniberitaasyik<BR>
tulis "<h2>Berita Selanjutnya</h2>
tulis "iniberitaasyik<BR>
tulis "iniberitaasyik<BR>
tulis "Prospek software di Indonesia <BR>
ISIWEB_AKHIR
KERTASWEB_AKHIR
TAMAT
```

Figure 27. Default HTML style in BAIK.

Figure28. Example code to produce and use the standard HTML layout in a web page.

A menu in left side area of web page, can be created using keywords Kertasweb_Menu_Awal for beginning and Kertasweb_Menu_Akhir for ending. The text for Copyright area is provided by Kertasweb_Hakcipta. The main content is tagged by keywords: Isiweb_Awal and Isiweb_Akhir. We use Isiweb_Titel to create Title of Web Content.

Figure 29. BAIK in multitier web system.
In this paper researcher introduce the basic design of BAIK, a scripting language which syntax is in Indonesian for multi-tier web development. The basic parsing method, parameter handling and language construction for web programming are explained. The main performance of this design depends on variable handling using Binary Search Tree which has average complexity $O(\log n)$ [6].

BAIK scripting language is still far from modern programming language. However, a strong fundamental for developing software using programming language with Indonesian syntax has been established. Currently, implementation of BAIK language is adequate enough for mathematical computation and data presenting or manipulation in web business application.

For future work, researcher will first focus on adding language base, especially for multi-thread execution, polymorphism and inheritance for object oriented programming. The second step is providing modular design of calling external library function, so it is easy to add new function capability into BAIK language without changing the core design. Improving LF by adopting external open source libraries is a must. Currently, BAIK language supports basic TCP/IP network programming, database connection using Mysql[11] library and creating picture file in PNG and JPG format using GD graphic library[12]. In near future, researcher plan to add functions of simple network management protocol, secure socket layer and other databases supports.

Lastly, adopting Apache module library in order to support client side HTML script programming and to make design of database connection pool is the next step of BAIK implementation in multitier web system. Improving the performance of simultaneous database access for web system will be the main issue. The source code, executable program and document in Indonesian of BAIK language introduced in this paper are available on the following website:
http://sourceforge.net/projects/baik/

4. Conclusion

Acknowledgement

Haris Hasanudin was born in Indramayu, West Java – Indonesia in 1974. He received BSc and MSc degree in Computer Science from Gunma University, Japan in 1998 and 2000 respectively. He is a senior system analyst at Mitsubishi Information Network Corporation, Japan. He became a Member (M) of IEEE since 2001.
Reference