ISSN 1410-9557 Volume III, Nomor 5, 2002

HASIL PENELITIAN UPN "VETERAN" YOGYAKARTA

Diterbitkan oleh :
Lembaga Penelitian UPN "Veteran" Yogyakarta
2002

Volume III, Nomor 5

Cetakan Kelima, 2002

Penerbit :

UPN "Veteran" Yogyakarta Press
Jl. Lingkar Utara Condongcatur, Yogyakarta 55283
Telp (0274) 486733, Fax (0274) 486400

Pencetak :

UPT Percetakan UPN "Veteran" Yogyakarta

ISSN 1410-9557

ISSN : 1410-9557 Volume III, Nomor 5, 2002

HASIL PENELITIAN
UPN "VETERAN" YOGYAKARTA

Diterbitkan oleh :
Lembaga Penelitian UPN "Veteran" Yogyakarta
2002

DARI REDAKSI

Journal hasil penelitian UPN "Veteran" Yogyakarta kembali tampil dengan hasil penelitian dalam bidang ilmu dan teknologi kebumian seperti Batas Kuarter Berdasarkan Analisis Nannofosil Daerah Rawa Bendungan, Cilacap, Jawa Tengah.

Di bidang pertanian ditampilkan Uji Aplikasi Secara Fumigasi Ekstrak Kunyit Terhadap Mortalitas *Sitophilus Oryzae* (Coleoptera : Curculionidae) : Pengaruh Volume Ruang dan Konsentrasi Ekstrak, Penyaringan Toleransi Salinitas Beberapa Kultivar Padi pada Fase Perkecambahan, Pengaruh Persentase Defoliasi Daun pada Beberapa Fase Tumbuh terhadap pertumbuhan dan Hasil Kedelai.

Pada bidang Industri menampilkan Analisis dan Perancangan Ergonomis Sistem Perlintasan Kereta Api Guna Meminimalkan Resiko Terjadinya Tabrakan Kereta dengan Kendaraan, Analisis Peningkatan Mutu Produk Susu *Full Cream Milk Powder* dengan Menggunakan Metode *Quality Function Deployment* (Studi Kasus Di PT Sari Husada Yogyakarta), Perancangan Transformasi Kontek Bahasa Bebas, untuk Pengajaran Teori Bahasa dan Otomata.

Redaksi tidak hanya memuat hasil penelitian yang dibiayai UPN "Veteran" Yogyakarta saja, tetapi juga menerima hasil penelitian lain yang belum pernah dipublikasikan.

Akhirnya redaksi mengucapkan terima kasih kepada semua fihak yang telah membantu terwujudnya penerbitan "Hasil Penelitian" ini. Semoga publikasi ini dapat hadir dihadapan pembaca setiap semester sekali.

Yogyakarta, Mei 2002

Redaksi

PERANCANGAN TRANSFORMASI KONTEK BAHASA BEBAS, UNTUK PENGAJARAN TEORI BAHASA DAN OTOMATA [1)]

(Free Grammar Context Transformation Design for Teaching of Language and Automata Theory)

Paryati, ST [2)]

ABSTRACT

In the implementation of learning-teaching process in Language and Automata Theory lecturing, there is often some difficulties to the students to understand the materials given. It is especially for anything abstract or difficult to view the real form. A teaching equipment of language and automata theory can make easy and speed up the delivery a lecturing topic because it gives a visualization. The language and automata theory is a subject contains many materials and theories that are directly difficult to understand by the students. One of the materials is about free grammar context, Chomsky Normal Form, and Greibach Normal Form. By the supporting of teaching equipment, it is hoped that the students will be more easily understand the transformation mechanism happen in every step of free grammar context.

INTI SARI

Dalam pelaksanaan proses belajar mengajar pada perkuliahan Teori Bahasa dan Otomata, sering terdapat kesulitan pada peserta didik untuk memahami materi-materi yang disampaikan. Terutama apabila terdapat hal-hal yang cenderung abstrak dan tidak mudah dipahami wujud riilnya. Suatu perangkat ajar teori bahasa dan otomata ini dapat mempermudah dan mempercepat penyampaian suatu topik perkuliahan karena memberikan visualisasi. Teori bahasa dan otomata adalah suatu mata kuliah yang memuat banyak kajian dan teori yang sulit dipahami secara langsung oleh peserta didik. Diantara materinya adalah mengenai kontek free grammar, Chomsky normal form, Greibach normal form. Dengan dukungan perangkat ajar ini diharapkan peserta didik akan lebih mudah memahami mekanisme transformasi yang terjadi pada setiap tahapan dalam context free grammar.

[1)] *Dibiayai Lembaga Penelitian UPN "Veteran" Yogyakarta*

[2)] *Staff Pengajar Jurusan Teknik Informatika UPN "Veteran" Yogyakarta*

I. PENDAHULUAN

1.1. Latar Belakang Masalah

Dalam pelaksanaan proses belajar mengajar pada perkuliahan, sering sekali terdapat kesulitan peserta didik untuk memahami materi yang disampaikan. Terutama bila terdapat hal-hal yang cenderung abstrak dan tidak mudah dipahami wujud riilnya. Suatu perangkat ajar dapat mempermudah dan mempercepat penyampaian suatu topik atau perkuliahan karena memberikan visualisasi dan simulasi, sehingga dapat menjembatani antara materi perkuliahan dengan peserta didik.

Teori Bahasa dan Otomata adalah suatu mata kuliah yang memuat banyak teori-teori dan kajian serta simbol-simbol yang agak sulit untuk dipahami secara langsung oleh peserta didik. Di antara materinya adalah mengenai Context Free Grammar, Chomsky Normal Form, Greibach Normal Form. Dengan dukungan perangkat ajar ini diharapkan peserta didik akan lebih mudah memahami mekanisme transformasi yang terjadi pada setiap tahapan dalam Context Free Grammar.

1.2. Perumusan Masalah

Dalam penelitian ini yang menjadi pokok permasalahan adalah **bagaimana mengembangkan suatu perangkat lunak yang mampu memberikan pemahaman yang lebih tepat dan mendalam kepada mahasiswa mengenai proses-proses transformasi yang terjadi pada Context Free Grammar.** Hal tersebut dapat dilakukan dengan melalui visualisasi dan simulasi pada setiap tahapan proses.

1.3. Tujuan Penelitian

Penelitian ini memiliki tujuan sebagai berikut :

a. Mengembangkan perangkat lunak ajar mata kuliah Teori Bahasa dan Otomata, khususnya materi Context Free Grammar.
b. Mempermudah dan meningkatkan pemahaman peserta didik pada materi perkuliahan tersebut.
c. Memberi kemudahan dalam memahami mekanisme transformasi yang terjadi pada setiap tahapan dalam Context Free Grammar.

1.4. Kontribusi Penelitian

Manfaat dari penelitian ini diharapkan dapat berguna sebagai perangkat ajar pengembangan perangkat lunak mata kuliah Teori Bahasa dan Otomata,

sehingga dapat meningkatkan kemampuan mahasiswa yang nantinya akan meningkatkan kualitas dari lulusan UPN "VETERAN" Yogyakarta.

II. LANDASAN TEORI

2.1 Hirarki Chomsky

Menurut Chomsky (Ullman, 1979), bahasa dapat diklasifikasikan menjadi empat kelompok:

a. Regular Grammar
b. Context Free Grammar
c. Context Sensitive Grammar
d. Unrestricted Grammar

Context Free Grammar

Merupakan bahasa yang berguna untuk mendeskripsikan ekspresi aritmatika, misalnya yang memiliki banyak kurung buka dan kurung tutup, serta struktur blok dalam bahasa pemrograman.

Tahapan proses Context Free Grammar.

Suatu Context Free Grammer dapat mengalami beberapa tahapan proses, sebagai berikut:

- Tidak ada produksi ε
- Tidak ada produksi unit
- Tidak ada simbol yang bersifat *useless*
- Penghilangan rekursif kiri
- Bentuk normal Chomsky
- Bentuk normal Greibach

Penghilangan produksi yang useless

Produksi yang *useful* memiliki syarat:

- Simbol x harus dapat muncul dalam sentensial form yang diturunkan dari simbol awal
- Simbol x tersebut harus diturunkan menjadi rangkaian simbol-simbol terminal

Jika kedua syarat tersebut tidak dipenuhi, maka x dikatakan simbol yang tidak bermanfaat (useless)

Prosedur penghilangan simbol yang useless:

a. Penentuan simbol-simbol variabel yang tidak menghasilkan simbol-simbol terminal
   1. Kumpulkan semua aturan yang ruas kanannya terdiri hanya dari simbol-simbol terminal

2. Dari tahap 1 ini kita ketahui daftar simbol-simbol setiap variabel yang dapat menghasilkan string terminal
3. Dari aturan produksi yang tersisa dari tahap 1 periksa apakah simbol-simbol pada ruas kanannya muncul dalam daftar simbol yang dihasilkan dari tahap 2.

b. 1. Kumpulkan semua aturan produksi yang ruas kirinya = simbol awal
2. Dari list no 1 kumpulkan variabel-variabel yang muncul di ruas kanan (List R)
3. Untuk setiap variabel yang muncul di R ambil aturan produksi yang ruas kirinya X. Tambahkan simbol-simbol variabel pada ruas kanan aturan produksi tersebut ke dalam R
4. Ulangi langkah 3 sampai semua variabel di R telah diperiksa
5. Aturan-aturan produksi yang ruas kirinya tidak muncul di R dapat dihilangkan

Penghilangan produksi unit

1. Kumpulkan semua aturan produksi non unit
2. Dari kumpulan aturan produksi unit yang tersisa dalambentuk $A \rightarrow B$
   a. Dari tahap 1 diketahui seluruh aturan produksi yang ruas kirinya B misalkan $B \rightarrow \gamma 1|\gamma 2|\gamma 3|...|\gamma n$ $\quad \gamma i \in (V \cup T)^*$
   b. Ganti produksi $A \rightarrow B$ dengan $A \rightarrow \gamma 1|\gamma 2|...|\gamma n$

Penghilangan produksi ε (dalam bentuk $A \rightarrow \varepsilon$)

Syarat: bahasa yang didefinisikan oleh aturan produksinya merupakan bahasa yang tidak menghasilkan ε

1. Kumpulkan semua aturan produksi yang menghasilkan ε (ruas kanan =ε). Dari tahap ini diperoleh daftar variabel yang menghasilkan ε (List K)
2. Periksa aturan produksi yang masih ada. Jika ruas kanannya mengandung variabel yang ada pada daftar dari no (1) Tandai variabel tersebut,. Jika ternyata dihasilkan produksi yang ruas kanannya tidak memiliki simbol yang bertanda tambahkan ruas kiri dari aturan produksi pada list K
3. Lakukan tahap 2 sampai tidak ada lagi simbol yang ditambahkan ke K

Penghilangan rekursif kiri

Secara umum kumpulan aturan produksi yang mengandung aturan produksi rekursif kiri

$A \rightarrow A\alpha 1|A\alpha 2|...|A\alpha n|\gamma 1|\gamma 2|...|\gamma m$

$\gamma i, \alpha i \in (V \cup T)^*$

dapat diubah menjadi kumpulan aturan produksi yang tidak rekursif kiri

A→γ1|γ2|γm|γA'|γ2A'|...|γmA'
A'→α1α'|α2A'|...|αnA'|α1|α2|...|αn
yaitu dengan cara:

1. Menambahkan simbol variabel baru (A')
2. Mengubah (m+n) aturan produksi semula, menambahkan sebanyak 2n+m aturan produksi baru
3. Menghilangkan n aturan produksi asal yang rekursif kiri

Bentuk Chomsky Normal Form

Bentuk CNF didapatkan dengan cara
- Menghilangkan produksi ε
- Menghilangkan produksi unit
- Tidak ada simbol yang bersifat useless

Algoritma formal pengubahan CFG ke dalam CNF:

1. Kumpulkan semua aturan produksi yanng ruas kanannya terdiri hanya dari satu simbol terminal
2. Untuk produksi-produksi dalam bentuk
   A→X1,X2,...,Xm m>=2
   Jika Xi adalah simbol terminal a
   - Buat produksi baru Ca →a
   - Ganti Xi pada aturan produksi di atas dengan Ca
3. Setelah tahap 2 di atas dikerjakan, maka seluruh aturan produksi yang tersisa ruas kanannya terdiri dari simbol-simbol variabel saja
   i) Produksi yang ruas kanannya terdiri dari dua variabel langsung dapat disertakan sebagai produksi baru
   ii) Untuk produksi yang terdiri dari >2 variabel
   A→B1B2...Bm m>2

III. METODOLOGI PENELITIAN

Secara garis besar langkah-langkah pemecahan masalah dibagi dalam beberapa tahap :

3.1. Identifikasi Masalah

Adalah tahap untuk melihat dan mengklasifikasikan permasalahan sesuai dengan tujuan penelitian.

3.2. Studi Pustaka

Studi pustaka perlu dilakukan untuk mempelajari buku-buku referensi yang berkaitan dengan pokok permasalahan yang akan dibahas dan metode yang dipakai dalam menganalisa pokok permasalahan.

3.3. Tahap Perancangan.

3.4. Tahap Pemrograman.
3.5 Tahap Pengujian Sistem.

IV. ANALISIS DAN PERANCANGAN

4.1 Mengidentifikasi Permasalahan.

Membuat program dalam bahasa C atau C++ yang memiliki kemampuan untuk:
1. Membaca deskripsi suatu tata bahasa *context-free,* dan
2. Mengubah deskripsi tata bahasa tersebut ke dalam bentuk lain yang memenuhi kriteria tertentu seperti:

- Tidak ada produksi ε
- Tidak ada produksi unit
- Tidak ada simbol yang bersifat *useless*
- Penghilangan rekursif kiri
- Bentuk normal Chomsky
- Bentuk normal Greibach

Kriteria ini akan ditentukan oleh pemakai program melalui menu yang disediakan oleh program.

4.2 Aturan Penulisan Bahasa Context Free

Deskripsi tata bahasa context free akan diberikan dalam bentuk arsip tekstual dengan aturan penulisan sbb:

1. Arsip deskripsi tata bahasa akan terdiri dari 4 (empat) bagian, masing-masing untuk menerangkan variabel, simbol terminal, simbol awal dan aturan-aturan produksi.
2. Setiap bagian akan didahului dengan header yang menyatakan jenis bagian tersebut. Header yang mungkin muncul adalah salah satu dari empat string berikut: **$VARIABEL, $TERMINAL, $AWAL**, atau **$PRODUKSI**. Keempat bagian ini dapat dituliskan dalam urutan yang tidak ditentukan, tetapi **$PRODUKSI** selalu dituliskan pada bagian terakhir sehingga pada saat aturan produksi dibaca, jenis setiap simbol sudah diketahui.
3. Setiap simbol variabel dapat dituliskan sebagai string alfabet
4. Setiap simbol terminal merupakan satu karakter tunggal dan akan dituliskan langsung sebagai karakter tersebut, kecuali untuk simbol-simbol terminal khusus berikut ini, koma (','), backslash ('\'), dan titik koma (';'). Simbol-simbol terminal khusus ini akan dituliskan dengan didahului tanda bakslash seperti yang tertera pada tabel .
5. Simbol string kosong (ε) akan dituliskan sebagai '\0'
6. Penulisan simbol variabel dan terminal akan dipisahkan oleh tanda koma (',') dan diakhiri oleh tanda titik koma (';')

7. Ruas kiri dan ruas kanan aturan produksi tidak dipisahkan oleh simbol apapun dengan asumsi bahwa simbol pertama adalah ruas kiri dari aturan produksi
8. Akhir dari sebuah aturan produksi akan ditandai dengan tanda titik koma (';').

Simbol Terminal	Karakter	Penulisan
Koma	,	\,
Backslash	\	\\
titik koma	;	\;

4.3 Perancangan

Pendefinisian struktur data mempunyai peranan yang penting dalam pembuatan program ini. Struktur yang mengkodekan simbol-simbol ke dalam angka yang memudahkan proses pembuatan algoritma. Struktur data yang dibuat harus mampu menangani kemungkinan-kemungkinan yang terjadi dalam setiap proses.

Struktur data program

```
#define end_mark      100
#define e_mark        98
#define max_var       20      /* indeks maximal variabel */
#define min_terminal 21
#define max_terminal 40       /* indeks maximal terminal  */
#define max_prod       100      /* jumlah maximal produksi
untuk satu variabel */
#define max_item      15       /* jumlah item --> variabel,
terminal atau tanda or */

typedef char char_arr[20];
typedef struct
      {
        char_arr first;                    /* symbol awal */
        char_arr variable[max_var];        /* symbol variabel */
        char terminal[max_terminal];       /* symbol terminal */
      } symbol_rec;

/* record produksi */
typedef struct
      {
       short production[max_prod][max_item];     /* produksi
*/
       short list_var[max_prod];                  /* daftar
variabel yang berproduksi */
      } product_rec;
```

```
// Deklarasi klass untuk CFG
class Tcfg
    {
     private:
      short sum_of_variable,          /* jumlah variabel */
          sum_of_terminal,          /* jumlah terminal */
          sum_of_product;           /* jumlah produksi */

     public:
      symbol_rec symbol;     /* symbol variabel, terminal,
dan awal */
      product_rec product; /* produksi */

      Tcfg();
      void set_cfg(short sum_var,short sum_terminal,short
      sum_product,
                symbol_rec sym, product_rec prod);
      void get_cfg(short &sum_var,short &sum_terminal,short
      &sum_product,
                 symbol_rec &sym, product_rec &prod);
      boolean is_var(short row,short col);
      /* true jika st merupakan variabel */
      short get_amount_var();
      short get_amount_terminal();
      short get_amount_product();
      boolean is_any_e();
      boolean is_any_recursive();
      boolean is_any_unit();
     };

typedef boolean list_1_terminal_arr[max_prod];
typedef short list_new_var_arr[max_terminal];
typedef boolean list_select_arr[max_terminal];

// Deklarasi class untuk CNF
class Tcnf
  {
    private:
      short sum_of_variable1,              /* jumlah variabel
*/
                sum_of_terminal1,                    /* jumlah
terminal */
          sum_of_product1;                 /* jumlah produksi
*/
      short sum_of_variable2,              /* jumlah variabel
*/
```

```
            sum_of_terminal2,                    /* jumlah terminal
*/
            sum_of_product2;                     /* jumlah produksi
*/
        list_1_terminal_arr list_1_terminal;
        list_new_var_arr list_new_var;
        list_select_arr list_select;
        short count_amount_var(short row);
        /* I.S. : produksi hanya terdiri dari variabel saja*/
        /* F.S. : menghitung jumlah variabel */
        boolean is_var(short row,short col);
        /* cek apakah variabel atau bukan */
        boolean is_1_terminal(short row);
        /* cek apakah produksi terdiri dari satu terminal */
        short set_new_var(short row,short col);
        /* mengisi variabel-variabel baru */
        short get_new_var();
        /* I.S. : row,col merupakan posisi suatu terminal */
        void set_new_symbol_var(short id);
        /* membuat simbol variabel baru */

      public:
        symbol_rec symbol1; /* symbol variabel, terminal, dan
awal */
        symbol_rec symbol2; /* symbol variabel, terminal, dan
awal */
        product_rec product1; /* produksi */
        product_rec product2; /* produksi hasil */
        Tcnf();
        void set_data(short sum_var,short sum_terminal,short
        sum_product,symbol_rec  sym, product_rec prod);
      void get_data(short &sum_var,short &sum_terminal,short
        &sum_product,symbol_rec &sym, product_rec &prod);
        void cfg_2_cnf();
  };


/* definisi struktur data untuk Greibach */
typedef struct
{
 int jumprod;
 int prod[6][10];
}elemen;

typedef elemen matrik[10][10];

class Greibach
```

```
{
   private:
     short sum_of_variable,              /* jumlah variabel
*/
           sum_of_terminal,              /* jumlah terminal
*/
           sum_of_product;               /* jumlah produksi
*/
     symbol_rec symbol;                          /* symbol
variabel,   terminal, dan awal */
     product_rec product;                  /* produksi */

    public:
     Greibach();
     void set_data(short sum_var,short sum_terminal,short
     sum_product,symbol_rec  sym, product_rec prod);
    void get_data(short &sum_var,short &sum_terminal,short
     &sum_product,symbol_rec &sym, product_rec &prod);
     void cfg_2_gnf();
  };

//======------------------------------------======
//= Deklarasi class untuk menghilangkan simbol =
//=              yang useless                  =
//======------------------------------------======

typedef boolean list_prod_terminal_arr[max_prod];
typedef boolean list_r_arr[max_var];
typedef  boolean list_var_terminal_arr[max_var];
class Not_any_useless
   {
     private:
     class Tcfg cfg;
     symbol_rec symbol;     /* symbol variabel, terminal,
dan awal */
     product_rec product;  /* produksi */
     short sum_of_variable,                    /* jumlah
variabel */
     sum_of_terminal,                /* jumlah terminal */
     sum_of_product;
     list_prod_terminal_arr list_prod_terminal;
     list_var_terminal_arr  list_var;
     list_r_arr list_r;
     boolean is_all_terminal(short row);
     /* true jika suatu produksi ruas kanannya terminal
semuanya */
     boolean is_right_terminal(short row);
```

```
      /* true jika satu produksi ruas kanannya dapat
menghasilkan
         terminal semua */
      boolean is_first(short row);
      /* true jika ruas kiri produksi merupakan symbol awal
*/
        void Not_any_useless::check_1_var();
        // mengecek jika sebuah ruas kanan hanya satu
variabel apakah     //masuk
        // dalam list r

      public:
      Not_any_useless();
      void set_cfg(Tcfg new_cfg);
      void          get_data(short          &sum_var,short
&sum_terminal,short     &sum_product,
                symbol_rec &sym, product_rec &prod);
      void remove_useless();
    };

//======------------------------------======
//= Deklarasi class untuk menghilangkan =
//=          produksi unit              =
//======------------------------------======

typedef boolean tab_unit_arr[max_prod];
class Not_any_unit
  { private:
      Tcfg cfg;
      symbol_rec symbol;      /* symbol variabel, terminal,
dan awal */
      product_rec product;    /* produksi */
      short sum_of_variable,                    /* jumlah
variabel */
      sum_of_terminal;       /* jumlah terminal */
      short sum_of_product;
      tab_unit_arr tab_unit;
      boolean is_any_unit();
      /* bernilai true jika masih ada produksi unit */

    public:
        Not_any_unit();
      boolean is_product_unit(short row);
      /* bernilai true jika produksi menghasilkan unit */
      void set_cfg(Tcfg new_cfg);
     void get_data(short &sum_var,short &sum_terminal,short
      &sum_product,symbol_rec &sym, product_rec &prod);
```

```
    void remove_unit();
  };

//======------------------------------======
//= Deklarasi class untuk menghilangkan =
//=            produksi e                =
//======------------------------------======

const int max_map   = 4;
const int max_combi = 31;
typedef boolean list_e_arr[max_var];
typedef boolean map_kombinasi_arr[max_combi][max_map];
class Not_any_e
   { private:
     Tcfg cfg;
     symbol_rec symbol;      /* symbol variabel, terminal,
dan awal */
     product_rec  product; /* produksi */
     short sum_of_variable,                    /* jumlah
variabel */
     sum_of_terminal,       /* jumlah terminal */
     sum_of_product;
     list_e_arr  list_e;
     map_kombinasi_arr  map_kombinasi;
     boolean is_product_e(short row);
     boolean is_prod_in_r(short row);
     /* apakah produksi semuanya variabel dan terdapat di
list R */
     short count_item_in_r(short row);
     /* menghitung jumlah variabel yang merupakan anggota
list r      pada
       suatu produksi */
     void fill_map(short range);
     /*  mengisi  peta  kombinasi  berdasarkan  jumlah
kombinasi yang    terjadi */
     boolean is_item_in_e(short row,short col);
     /* apakah sebuah item terdapat di list e */
     boolean is_s_any_e(short  &indeks);
     /* check apakah s menghasilkan e atau tidak */

      public:
       Not_any_e();
     void set_cfg(Tcfg new_cfg);
     void          get_data(short          &sum_var,short
&sum_terminal,short     &sum_product,symbol_rec     &sym,
product_rec &prod);
     void remove_e();
```

```
    };

//=======----------------------------======
//= Deklarasi class untuk menghilangkan =
//=          produksi rekursif          =
//=======----------------------------======

typedef boolean list_not_r_arr[max_prod];
class Not_any_recursive
   { private:
      Tcfg cfg;
      symbol_rec symbol;     /* symbol variabel, terminal,
dan awal */
      product_rec product;   /* produksi */
      short sum_of_variable,                    /* jumlah
variabel */
      sum_of_terminal,      /* jumlah terminal */
      sum_of_product;
      list_not_r_arr  list_not_r;
      list_not_r_arr list_select;
      boolean is_prod_recursive(short row);
      /* check apakah dalam satu produksi ada rekursif atau
tidak */
      boolean is_block_recursive();
      void add_var();
      /* menambah variable baru ke dalam symbol */
      boolean is_all_select();
      /* true jika semua produksi pada CFG telah diperiksa
*/

       public:
        Not_any_recursive();
      void set_cfg(Tcfg new_cfg);
      void          get_data(short          &sum_var,short
&sum_terminal,short     &sum_product,symbol_rec      &sym,
product_rec &prod);
      void remove_recursive();
   };
```

V. KESIMPULAN

Implementasi dilakukan pada sistem operasi DOS 6.22 Kompilator Borlan C++ 2.0. Pertimbangan implementasi pada sistem operasi DOS, agar program simulasi mudah dijalankan pada mesin dengan kebutuhan sumber daya yang kecil. Selain itu ukurannya executable yang kecil memungkinkan untuk dijalankan hanya dengan menggunakan floppy saja, sehingga mempermudah

peserta didik untuk mempergunakanya. Program lebih mudah didistribusikan untuk keperluan pengajaran dan mudah diinstalasi tanpa melakukan perubahan setting pada software di komputer tujuan.

Penggunaan bahasa C++ yang telah memberikan dukungan pemrograman berorientasi objek, mempermudah pengembangan sotware ini. Dan dapat dilakukan perubahan atau peningkatan dengan lebih baik.

Selain penggunaan objec dan class sebagai lazimnya pemrograman berorientasi objek, pengembangan perangkat lunak ini dilakukan secara modular dengan memanfaatkan kemampuan pengelolaan file projek (PRJ) dalam borlan C, untuk mempermudah melakukan testing dan debugging.

DAFTAR PUSTAKA

Hopcroft, Ullman, 1979, *Introduction to Automata Theory, Languages, and Computation*, Addison Wesley

Pressman, Roger S., 1992, *Software Engineering A Practitioner's Approach Third Edition*, McGraw-Hill

Wirth, Niklaus, 1978, *Algorithms and Data Structure*, ETH Zurich